Cars4U Project

Definition of the problem

There is a huge demand for used cars in the Indian Market today. As sales of new cars have slowed down in the recent past, the pre-owned car market has continued to grow over the past years and is larger than the new car market now. Cars4U is a budding tech start-up that aims to find footholes in this market.

In 2018-19, while new car sales were recorded at 3.6 million units, around 4 million second-hand cars were bought and sold. There is a slowdown in new car sales and that could mean that the demand is shifting towards the pre-owned market. In fact, some car sellers replace their old cars with pre-owned cars instead of buying new ones. Unlike new cars, where price and supply are fairly deterministic and managed by OEMs (Original Equipment Manufacturer / except for dealership level discounts which come into play only in the last stage of the customer journey), used cars are very different beasts with huge uncertainty in both pricing and supply. Keeping this in mind, the pricing scheme of these used cars becomes important in order to grow in the market.

As a senior data scientist at Cars4U, you have to come up with a pricing model that can effectively predict the price of used cars and can help the business in devising profitable strategies using differential pricing. For example, if the business knows the market price, it will never sell anything below it.

Project Objective

1. Explore and visualize the dataset.
2. Build a linear regression model to predict the prices of used cars.
3. Generate a set of insights and recommendations that will help the business.

Assumptions

The used car data is a simple random sample from the population data.

About the data

used_cars_data.csv - contains information about used cars.

1. S.No. : Serial Number
2. Name : Name of the car which includes Brand name and Model name
3. Location : The location in which the car is being sold or is available for purchase Cities
4. Year : Manufacturing year of the car
5. Kilometers_driven : The total kilometers driven in the car by the previous owner(s) in KM.
6. Fuel_Type : The type of fuel used by the car. (Petrol, Diesel, Electric, CNG, LPG)
7. Transmission : The type of transmission used by the car. (Automatic / Manual)

8. Owner : Type of ownership
9. Mileage : The standard mileage offered by the car company in kmpl or km/kg
10. Engine : The displacement volume of the engine in CC.
11. Power : The maximum power of the engine in bhp.
12. Seats : The number of seats in the car.
13. New_Price : The price of a new car of the same model in INR Lakhs.(1 Lakh = 100, 000)
14. Price : The price of the used car in INR Lakhs (1 Lakh = 100, 000)

Exploratory Data Analysis

Import the Python libraries

In [238...

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
# import statsmodels.api as sm
import scipy.stats as stats
from sklearn.preprocessing import LabelEncoder
import copy
import statsmodels.api as sm

import warnings
import math
from sklearn.preprocessing import LabelEncoder   # import label encoder.used for One Ho

warnings.filterwarnings("ignore")

sns.set(color_codes=True)
%matplotlib inline
#%load_ext nb_black
```

In [238...

```
sns.set() #setting the default seaborn style for our plots
```

Read the data into the notebook

In [238...

```
df = pd.read_csv('used_cars_data.csv') # import the .csv file as a data frame
```

View the first and last 5 rows of the dataset

In [238...

```
df.head()
```

Out[238...

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26 km/
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19. km

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18 km
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20. km
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15 km

In [238…

```
df.tail()
```

Out[238…

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	M
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	First	
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	First	
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	First	
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Third	
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	First	

Understand the shape of the dataset.

In [238…

```
df.shape
```

Out[238…

```
(7253, 14)
```

Check the data types of the columns for the dataset.

In [239…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   S.No.              7253 non-null   int64
 1   Name               7253 non-null   object
 2   Location           7253 non-null   object
 3   Year               7253 non-null   int64
```

```
 4   Kilometers_Driven  7253 non-null   int64
 5   Fuel_Type          7253 non-null   object
 6   Transmission       7253 non-null   object
 7   Owner_Type         7253 non-null   object
 8   Mileage            7251 non-null   object
 9   Engine             7207 non-null   object
 10  Power              7207 non-null   object
 11  Seats              7200 non-null   float64
 12  New_Price          1006 non-null   object
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

- Dataset has 7253 rows and 14 columns.

Fixing the data types

- Name, Location, Fuel_Type, Transmission , Owner_Type : These are categorical variables , so we can convert these features directly into category datatype.
- Year and Seats are numerical discrete values, but in this case it is also considered categorical.
- Mileage, Engine, Power and New_Price are object datatype, but we should consider converting these to numerical. We will work on these further below.

`coverting "objects" to "category" reduces the space required to store the dataframe. It also helps in analysis`

In [239…

```python
# For now we can convert Categorical and Numerical to categoriy datatype
df["Name"]=df["Name"].astype("category")

df["Location"]=df["Location"].astype("category")
df["Fuel_Type"]=df["Fuel_Type"].astype("category")
df["Transmission"]=df["Transmission"].astype("category")
df["Owner_Type"]=df["Owner_Type"].astype("category")

df["Year"]=df["Year"].astype("category")
```

In [239…

```python
# Lets check again
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   S.No.              7253 non-null   int64
 1   Name               7253 non-null   category
 2   Location           7253 non-null   category
 3   Year               7253 non-null   category
 4   Kilometers_Driven  7253 non-null   int64
 5   Fuel_Type          7253 non-null   category
 6   Transmission       7253 non-null   category
 7   Owner_Type         7253 non-null   category
 8   Mileage            7251 non-null   object
 9   Engine             7207 non-null   object
 10  Power              7207 non-null   object
 11  Seats              7200 non-null   float64
 12  New_Price          1006 non-null   object
 13  Price              6019 non-null   float64
```

```
dtypes: category(6), float64(2), int64(2), object(4)
memory usage: 600.6+ KB
```

Observations

- Some of the 'object' types have been converted to 'category' types.
- By converting the 'object' types to 'category' types, we have reduced the memory usage from 793.4+ KB to 551.4+ KB.

Five point summary of continuous variables

In [239...

```
df.describe(include='all').T
```

Out[239...

	count	unique	top	freq	mean	std	min	25%	50%	75%	ma
S.No.	7253	NaN	NaN	NaN	3626	2093.91	0	1813	3626	5439	725
Name	7253	2041	Mahindra XUV500 W8 2WD	55	NaN	NaN	NaN	NaN	NaN	NaN	Na
Location	7253	11	Mumbai	949	NaN	NaN	NaN	NaN	NaN	NaN	Na
Year	7253	23	2015	929	NaN	NaN	NaN	NaN	NaN	NaN	Na
Kilometers_Driven	7253	NaN	NaN	NaN	58699.1	84427.7	171	34000	53416	73000	6.5e+0
Fuel_Type	7253	5	Diesel	3852	NaN	NaN	NaN	NaN	NaN	NaN	Na
Transmission	7253	2	Manual	5204	NaN	NaN	NaN	NaN	NaN	NaN	Na
Owner_Type	7253	4	First	5952	NaN	NaN	NaN	NaN	NaN	NaN	Na
Mileage	7251	450	17.0 kmpl	207	NaN	NaN	NaN	NaN	NaN	NaN	Na
Engine	7207	150	1197 CC	732	NaN	NaN	NaN	NaN	NaN	NaN	Na
Power	7207	386	74 bhp	280	NaN	NaN	NaN	NaN	NaN	NaN	Na
Seats	7200	NaN	NaN	NaN	5.27972	0.81166	0	5	5	5	1
New_Price	1006	625	4.78 Lakh	6	NaN	NaN	NaN	NaN	NaN	NaN	Na
Price	6019	NaN	NaN	NaN	9.47947	11.1879	0.44	3.5	5.64	9.95	16

- S.No is the row Serial Number for each car. This is not statistically usefull. We will have to drop this variable.
- Kilometers_Driven MAX value looks unreal 6.5 million kilometers is too much. this may be an outlier and could potentially affect any statistic for this variable.
- Price. not all rows hava a price value. it looks like it contains many nulls(NaN). we will have to look into these in the Data Pre-Processing.

Summary of categorical variables

In [239...

```
df.describe(include=["category"]).T
```

Out[239...

	count	unique	top	freq
Name	7253	2041	Mahindra XUV500 W8 2WD	55
Location	7253	11	Mumbai	949
Year	7253	23	2015	929
Fuel_Type	7253	5	Diesel	3852
Transmission	7253	2	Manual	5204
Owner_Type	7253	4	First	5952

- Year has many entries (23), we may consider analyzing this variable within ranges.
- Seats variable has 53 rows with null (NaN) value.

In [239...

```
df.Location.value_counts()
```

Out[239...

```
Mumbai        949
Hyderabad     876
Kochi         772
Coimbatore    772
Pune          765
Delhi         660
Kolkata       654
Chennai       591
Jaipur        499
Bangalore     440
Ahmedabad     275
Name: Location, dtype: int64
```

In [239...

```
df.Year.value_counts()
```

Out[239...

```
2015    929
2014    925
2016    886
2013    791
2017    709
2012    690
2011    579
2010    407
2018    361
2009    252
2008    207
2007    148
2019    119
2006     89
2005     68
2004     35
2003     20
2002     18
2001      8
2000      5
1998      4
1999      2
1996      1
Name: Year, dtype: int64
```

In [239...

```
df.Fuel_Type.value_counts()
```

Out[239...

```
Diesel      3852
```

```
Petrol      3325
CNG           62
LPG           12
Electric       2
Name: Fuel_Type, dtype: int64
```

In [239…
```
df.Transmission.value_counts()
```

Out[239…
```
Manual       5204
Automatic    2049
Name: Transmission, dtype: int64
```

In [239…
```
df.Owner_Type.value_counts()
```

Out[239…
```
First             5952
Second            1152
Third              137
Fourth & Above      12
Name: Owner_Type, dtype: int64
```

In [240…
```
df.Seats.value_counts()
```

Out[240…
```
5.0     6047
7.0      796
8.0      170
4.0      119
6.0       38
2.0       18
10.0       8
9.0        3
0.0        1
Name: Seats, dtype: int64
```

Dropping unnecessary variables

We will drop the following variables/columns:

- S.No. : This is a unique identifier of every column, it really does not provide any statistical value.
- New_Price : This column has too many NULL values (87%)

In [240…
```
df.drop([ "S.No." , "New_Price" ] , axis=1 , inplace=True)
```

Check for missing values

In [240…
```
df.isna().sum()   #null value check
```

Out[240…
```
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 2
Engine                 46
Power                  46
Seats                  53
Price                1234
dtype: int64
```

Null values (out of 7253 rows)

- Mileage 2
- Engine 46
- Power 46
- Seats 53
- Price 1234

Cleanup variables

The following variables are identified as 'object' because they have characters that is affecting the numeric values. These can be cleanup in order for Python to recognize them as numeric.

- Mileage. The numerical value for this column has suffixes 'km/kg' and 'kmpl'
- Engine. The numerical value for this column has suffixes 'CC'
- Power. The numerical value for this column has suffixes 'bhp'

In [240…

```
def MileageToNum( mileage_val ):
    if not isinstance( mileage_val, float ):
        if isinstance( mileage_val, str ):
            return float(mileage_val.replace( "km/kg" , "").replace( "kmpl" , "").repla
        else:
            return np.nan
    else:
        return mileage_val

df["Mileage"] = df["Mileage"].apply( MileageToNum )
```

In [240…

```
def EngineToNum( engine_val ):
    if not isinstance( engine_val, float ):
        if isinstance( engine_val, str ):
            return float(engine_val.replace( "CC" , "").replace( " " , "") )
        else:
            return np.nan
    else:
        return engine_val

df["Engine"] = df["Engine"].apply( EngineToNum )
```

In [240…

```
def PowerToNum( power_val ):
    if not isinstance( power_val, float ):
        if isinstance( power_val, str ) and not power_val.startswith("null") :
            return float(power_val.replace( "bhp" , "").replace( " " , "") )
        else:
            return np.nan
    else:
        return power_val

df["Power"] = df["Power"].apply( PowerToNum )
```

In [240…

```
df.head()
```

Out[240…

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Eng
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.60	9

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Eng
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67	15
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.20	11
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77	12
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.20	19

Handle Nulls again

Fix missing values replacing with the Median

In [240...

```
# we will replace missing values in every column with its medain
medianFiller = lambda x: x.fillna(x.median())
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()
df[numeric_columns] = df[numeric_columns].apply(medianFiller,axis=0)
```

In [240...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype   
---  ------             --------------  -----   
 0   Name               7253 non-null   category
 1   Location           7253 non-null   category
 2   Year               7253 non-null   category
 3   Kilometers_Driven  7253 non-null   int64   
 4   Fuel_Type          7253 non-null   category
 5   Transmission       7253 non-null   category
 6   Owner_Type         7253 non-null   category
 7   Mileage            7253 non-null   float64 
 8   Engine             7253 non-null   float64 
 9   Power              7253 non-null   float64 
 10  Seats              7253 non-null   float64 
 11  Price              7253 non-null   float64 
dtypes: category(6), float64(5), int64(1)
memory usage: 487.3 KB
```

- We can observe that the variables Mileage , Engine and Power are now numeric variables

Finally, let convert Seats variable into categorical since this is a Numerical Discrete variable.

In [240...

```
df["Seats"]=df["Seats"].astype("category")
```

In [241...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
```

```
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype   
---  ------             --------------  -----   
 0   Name               7253 non-null   category
 1   Location           7253 non-null   category
 2   Year               7253 non-null   category
 3   Kilometers_Driven  7253 non-null   int64   
 4   Fuel_Type          7253 non-null   category
 5   Transmission       7253 non-null   category
 6   Owner_Type         7253 non-null   category
 7   Mileage            7253 non-null   float64 
 8   Engine             7253 non-null   float64 
 9   Power              7253 non-null   float64 
 10  Seats              7253 non-null   category
 11  Price              7253 non-null   float64 
dtypes: category(7), float64(4), int64(1)
memory usage: 438.1 KB
```

In [ ]:

EDA

Univariate analysison Numerical Variables

In [241…

```
def histogram_boxplot(feature , figsize=(15,10) , bins=None):
    """ Histogram and Boxplot combined
    feature: 1-d feature array
    figsize: size of figg.default (15,10)
    bins: number of bins.default None/auto
    """
    mean = feature.mean()
    median = feature.median()
    mode = feature.mode()

    f2, (ax_box2 , ax_hist2) = plt.subplots(nrows = 2, # num of rows of the subplot. gr
                                            sharex = True, # x-axis will be shared amon
                                            gridspec_kw = { "height_ratios": (.25 , .75
                                            figsize = figsize
                                           ) # create the 2 subplots

    sns.boxplot(feature , ax = ax_box2 , showmeans = True , color = 'red') # boxplot wi
    if bins:
        sns.distplot(feature , kde = False , ax = ax_hist2, bins = bins)
    else:
        sns.distplot( feature , kde = False , ax = ax_hist2 )
    ax_hist2.axvline( mean , color = 'green' , linestyle='-' , linewidth = 3 , label = 
    ax_hist2.axvline( median , color = 'yellow' , linestyle='-' , linewidth = 6 , label
    ax_hist2.axvline( mode[0] , color = 'black' , linestyle='-' , label = 'mode' ) # ad
    ax_hist2.legend()

    print( 'Mean:'+ str( mean ) )
    print( 'Median:'+ str( median ) )
    print( 'Mode:'+ str( mode[0] ) )
```

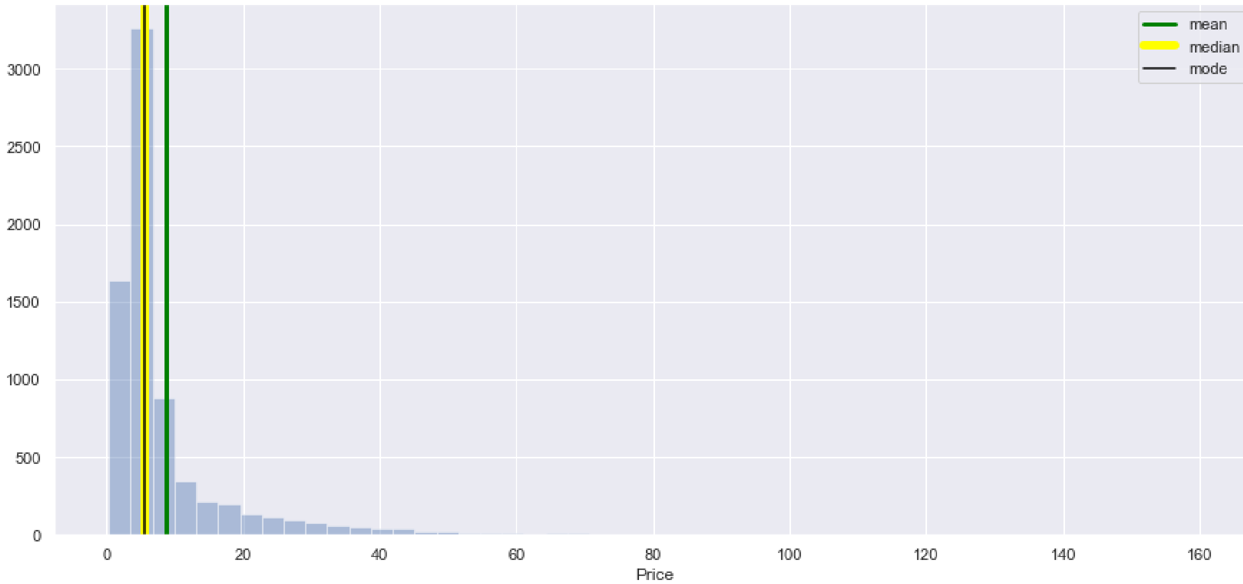
Price

In [241…

```
histogram_boxplot(df.Price)
```

```
Mean:8.826234661519258
```

```
Median:5.64
Mode:5.64
```



Observation

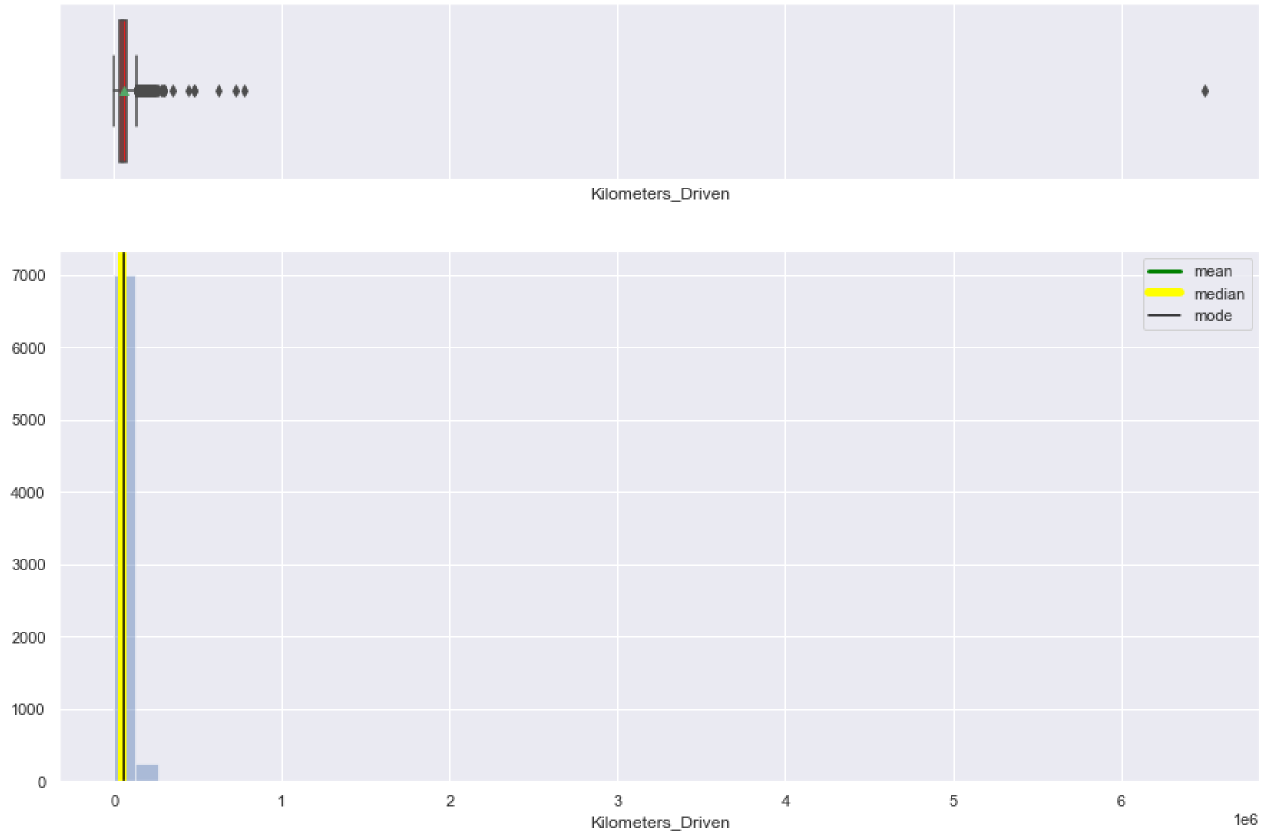
- Price is right skewed which means some cars have Price more than 50
- Mean Price is around 8.82
- There are many outliers on the right side of the whisker

Kilometers_Driven

In [241…

```
histogram_boxplot(df["Kilometers_Driven"])
```

```
Mean:58699.063146284294
Median:53416.0
Mode:60000
```



Observation

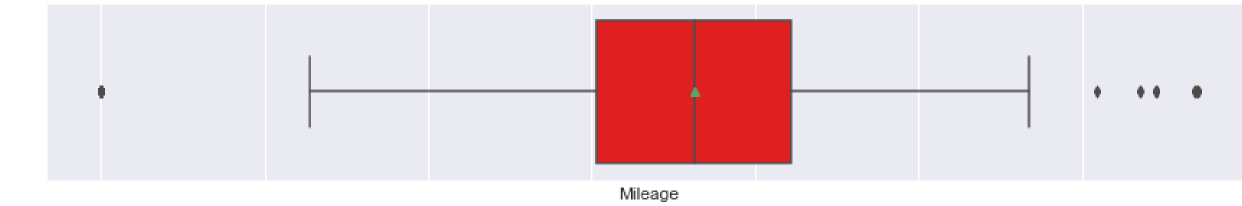
- Kilometers_Driven is right skewed
- Mean Kilometers_Driven is around 58699.06
- There are many outliers on the right side of the whisker

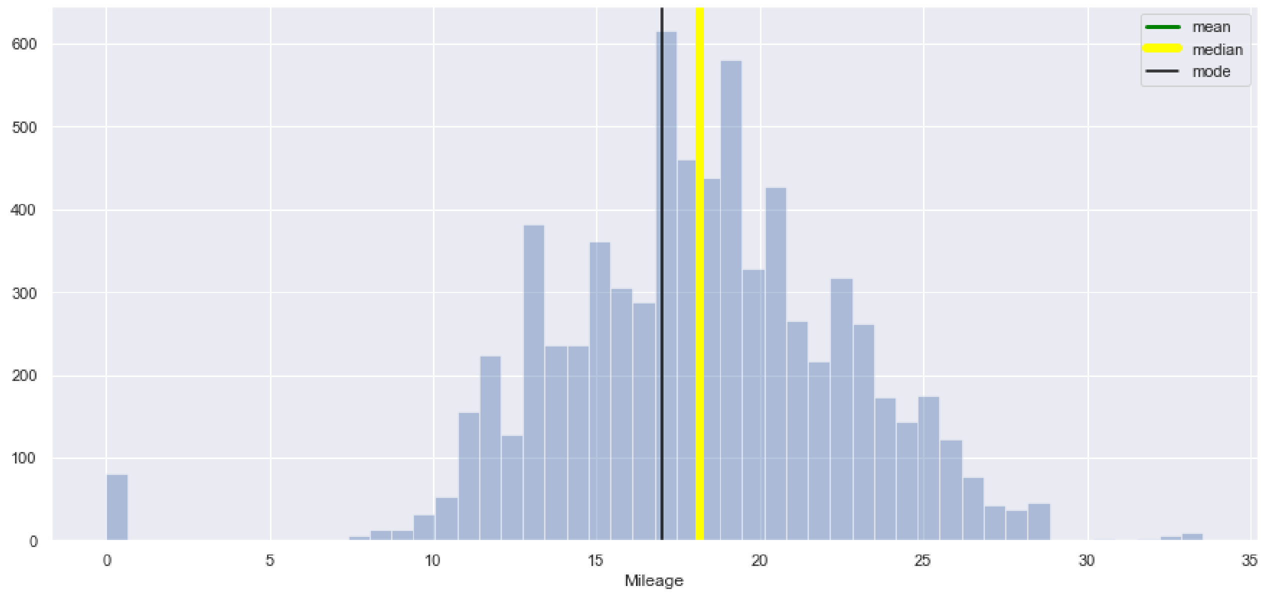
Mileage

In [241…

```
histogram_boxplot(df["Mileage"])
```

```
Mean:18.141585550806592
Median:18.16
Mode:17.0
```





Observation

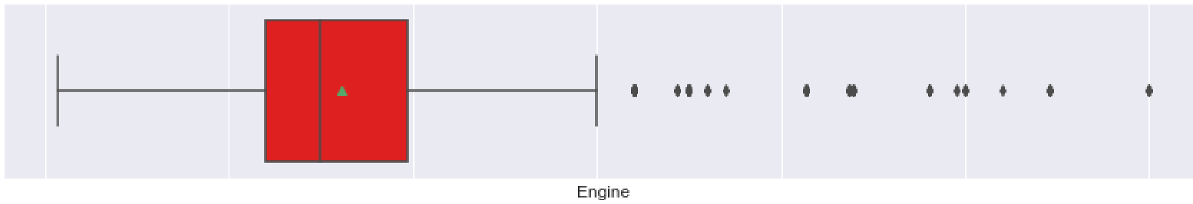
- Mileage mean is very close to the median which means that it is normally distributed
- Mean Mileage is around 18.14
- There are outliers on both sides of the distribution

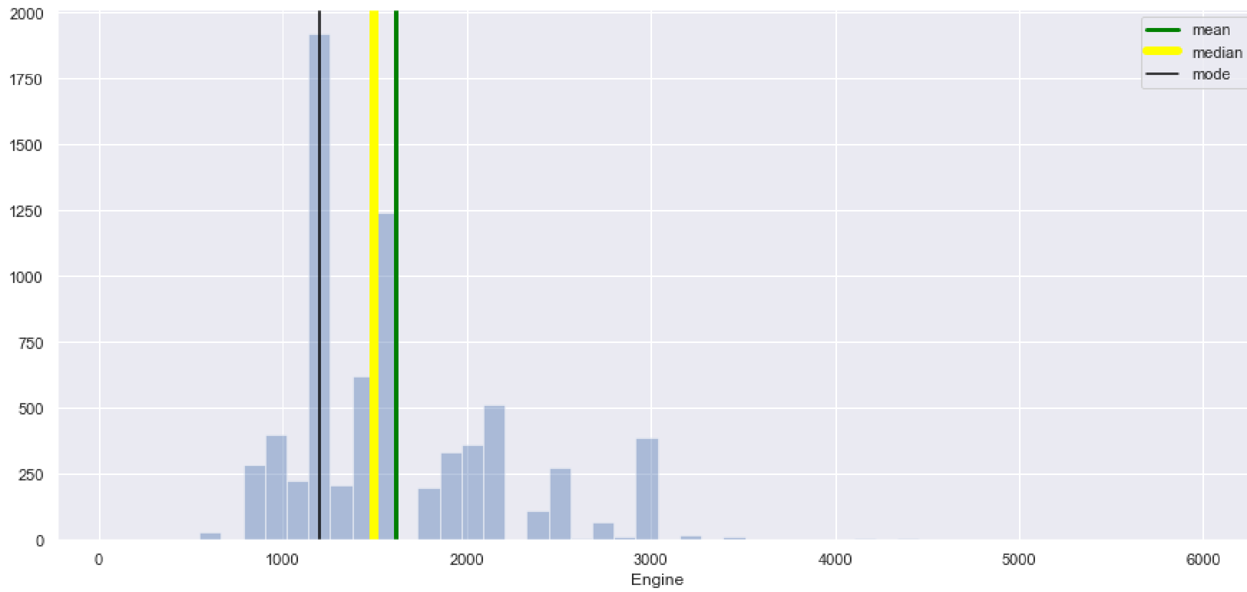
Engine

In [241…

```
histogram_boxplot(df["Engine"])
```

```
Mean:1615.7897421756516
Median:1493.0
Mode:1197.0
```





Observation

- Engine is right skewed which means some cars have Engine more than 3000CC
- Mean Engine is around 1616.57
- There are many outliers on the right side of the whisker

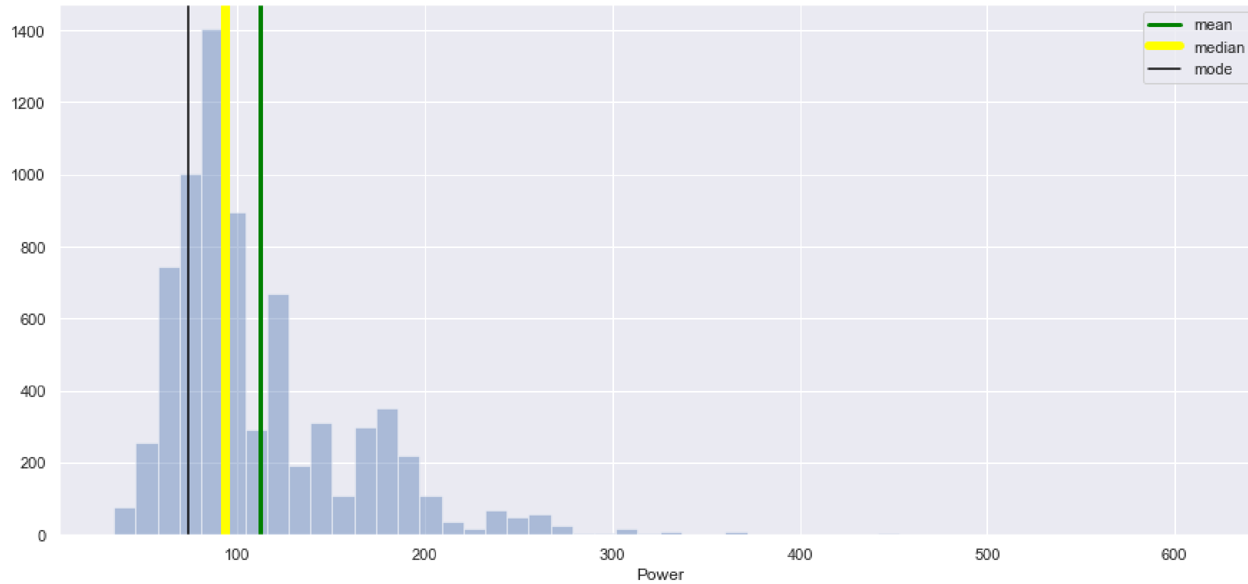
In [241…

```
### Power
```

In [241…

```
histogram_boxplot(df["Power"])
```

```
Mean:112.31244795257048
Median:94.0
Mode:74.0
```



Observation

- Power is right skewed which means some cars have Engine more than 250bhp
- Mean Power is around 112.76
- There are many outliers on the right side of the whisker

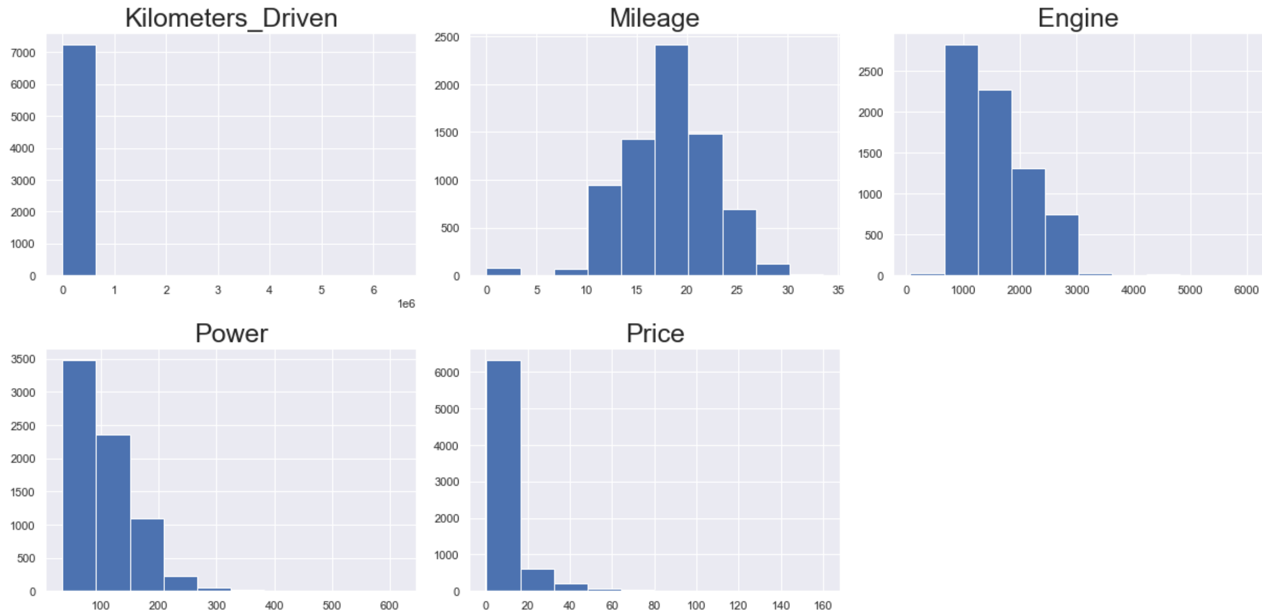
In [ ]:

```

```

Distribution of each numerical variable

In [241…

```
# Plot histogram of all plots
from scipy.stats import norm
all_col = df.select_dtypes(include=np.number).columns.tolist()
#all_col.remove('Year')
plt.figure(figsize=(17,75))

for i in range(len(all_col)):
    plt.subplot(18,3,i+1)
    plt.hist(df[all_col[i]])
    #sns.displot(df[all_col[i]], kde=True)
    plt.tight_layout()
    plt.title(all_col[i],fontsize=25)


plt.show()
```



Observation

- Mileage is normaly distributed
- All other variables are right skewed

Outlier Analysis in every numerical column

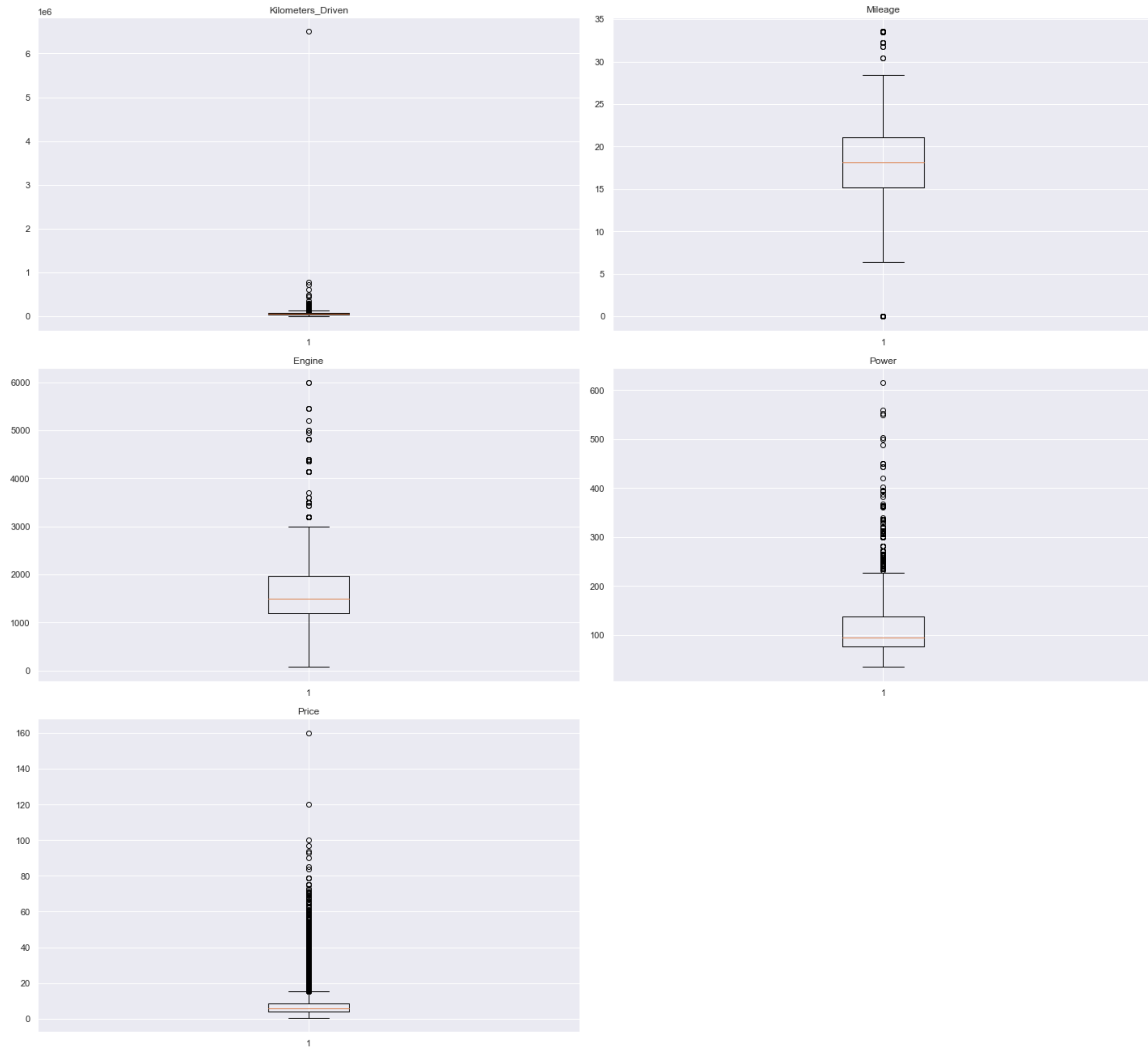
Fix missing values replacing with the Median

In [241…

```
# we will replace missing values in every column with its medain
medianFiller = lambda x: x.fillna(x.median())
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()
df[numeric_columns] = df[numeric_columns].apply(medianFiller,axis=0)
```

In [242…

```
plt.figure(figsize=(20,30))

for i, variable in enumerate( numeric_columns ):
    #print (variable)
                     plt.subplot(5,2,i+1)
                     plt.boxplot(df[variable],whis=1.5)
                     plt.tight_layout()
                     plt.title(variable)

plt.show()
```



- There are outliers in every variable
- Kilometers_Driven , Engine , Power , Price have upper outliers only
- Mileage has lower and upper outliers #### We will treat the outliers further ahead

In [ ]:

```

```

Univariate Analysis on Categorical Variables

Group Name into 'Make' value

Since the Name variable has 2041 unique value, we will try to group these into the 'Make' of the vehicle.We can try to split the Name and capture the first word which seems to be the vehicle 'Make' value.

In [242…

```
def NameSplitMake( name_val ):
    if isinstance( name_val, str ) and not name_val.startswith("null") :
        return name_val.split(' ')[0].upper()
    else:
        return np.nan
```

```
df["NameMake"] = df["Name"].apply( NameSplitMake ).astype("category")
```

In [242...
```
print(df['NameMake'].unique())
```

```
['MARUTI', 'HYUNDAI', 'HONDA', 'AUDI', 'NISSAN', ..., 'FORCE', 'BENTLEY', 'LAMBORGHINI',
'HINDUSTAN', 'OPELCORSA']
Length: 32
Categories (32, object): ['MARUTI', 'HYUNDAI', 'HONDA', 'AUDI', ..., 'BENTLEY', 'LAMBORG
HINI', 'HINDUSTAN', 'OPELCORSA']
```

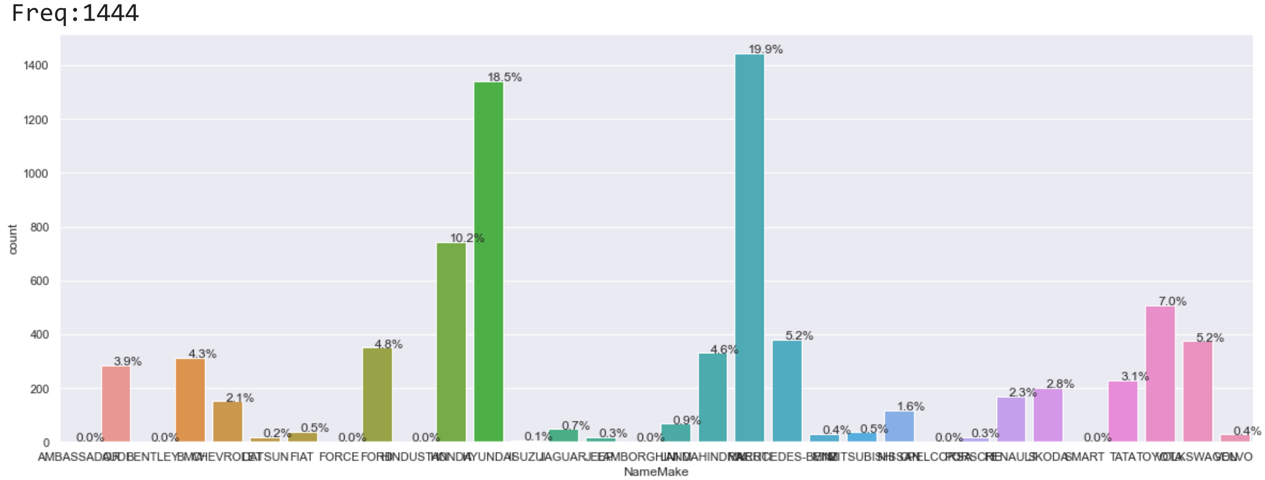
In [242...
```
def bar_count_pct( feature , figsize=(10,7) ):
    """
    feature : 1-d categorical feature array
    """
    mode = feature.mode()
    freq = feature.value_counts().max()

    plt.figure(figsize=figsize)

    ax = sns.countplot(feature )

    total = len(feature) # length of the column
    for p in ax.patches:
        percentage = '{:.1f}%'.format( 100 * p.get_height() / total ) # percentage of e
        x = p.get_x() + p.get_width() / 2 - 0.05 # width of the plot
        y = p.get_y() + p.get_height() # height of the plot
        ax.annotate( percentage , (x,y), size = 12) # annotate the percentage

    print( 'Top:'+ str( mode[0] ) )
    print( 'Freq:'+ str( freq ) )
```

NameMake

0 Name 7253 non-null category 1 Location 7253 non-null category 2 Year 7253 non-null category 4 Fuel_Type 7253 non-null category 5 Transmission 7253 non-null category 6 Owner_Type 7253 non-null category 10 Seats 7200 non-null category

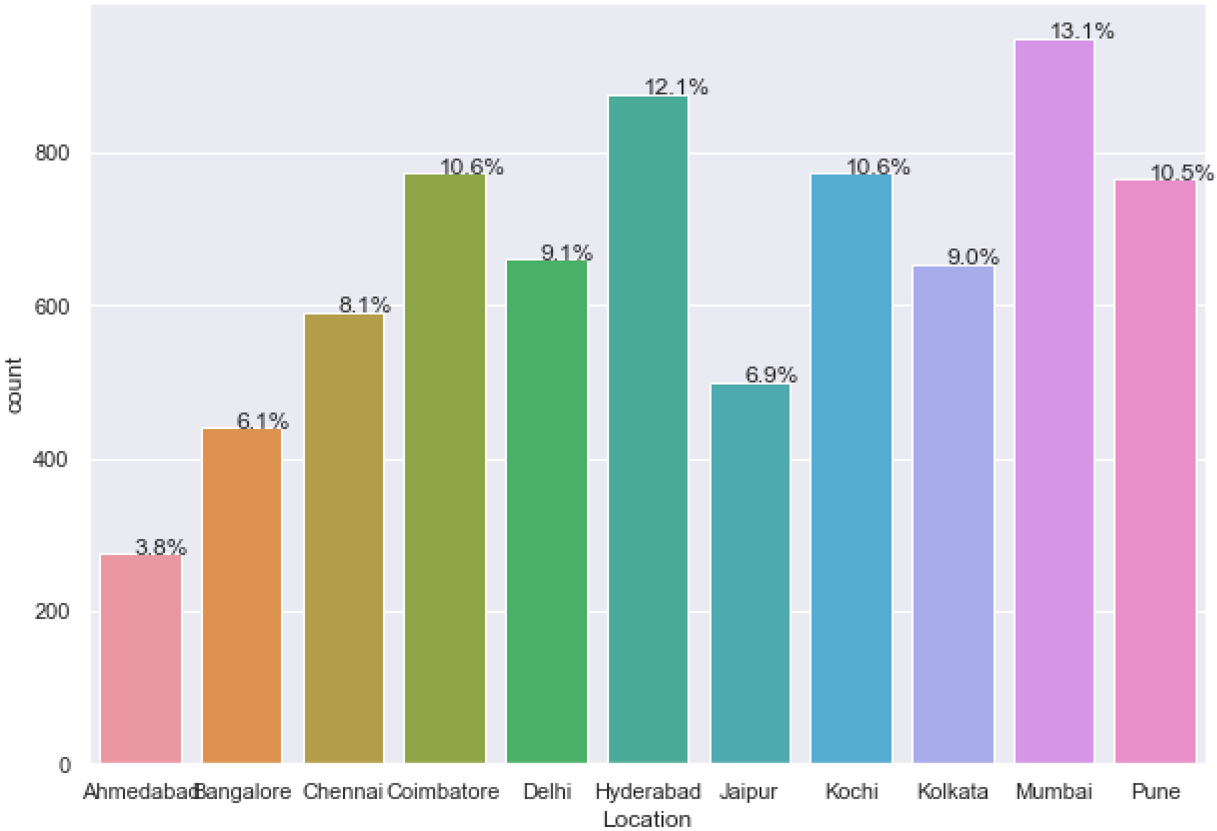
In [242...
```
bar_count_pct(df['NameMake'] , figsize=(20,7))
```

```
Top:MARUTI
Freq:1444
```



In [242...
```
bar_count_pct(df['Location'] )
```

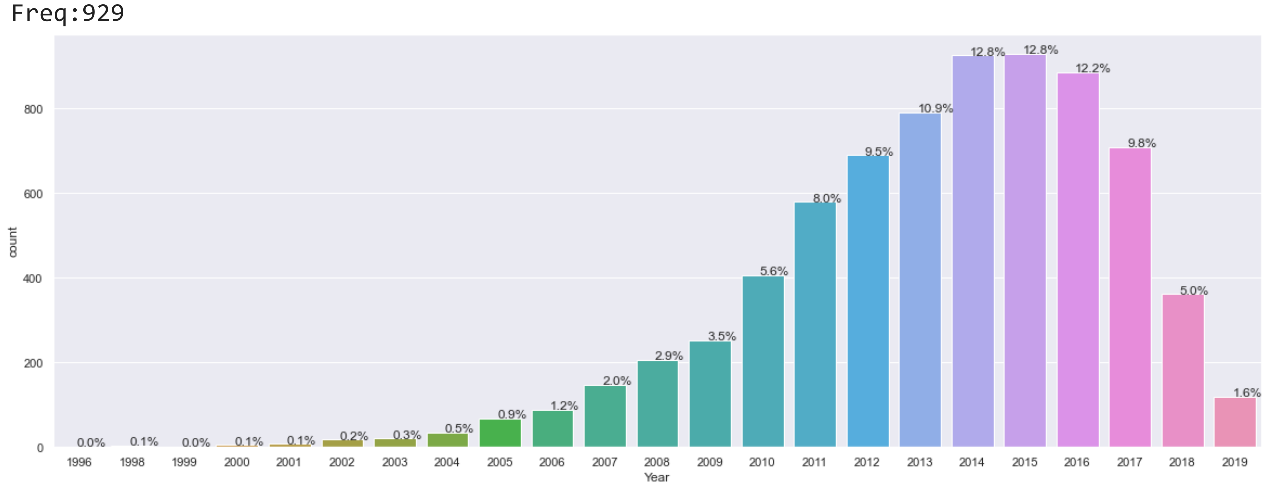
```
Top:Mumbai
Freq:949
```



In [242…

```
bar_count_pct(df['Year'] , figsize=(20,7) )
```

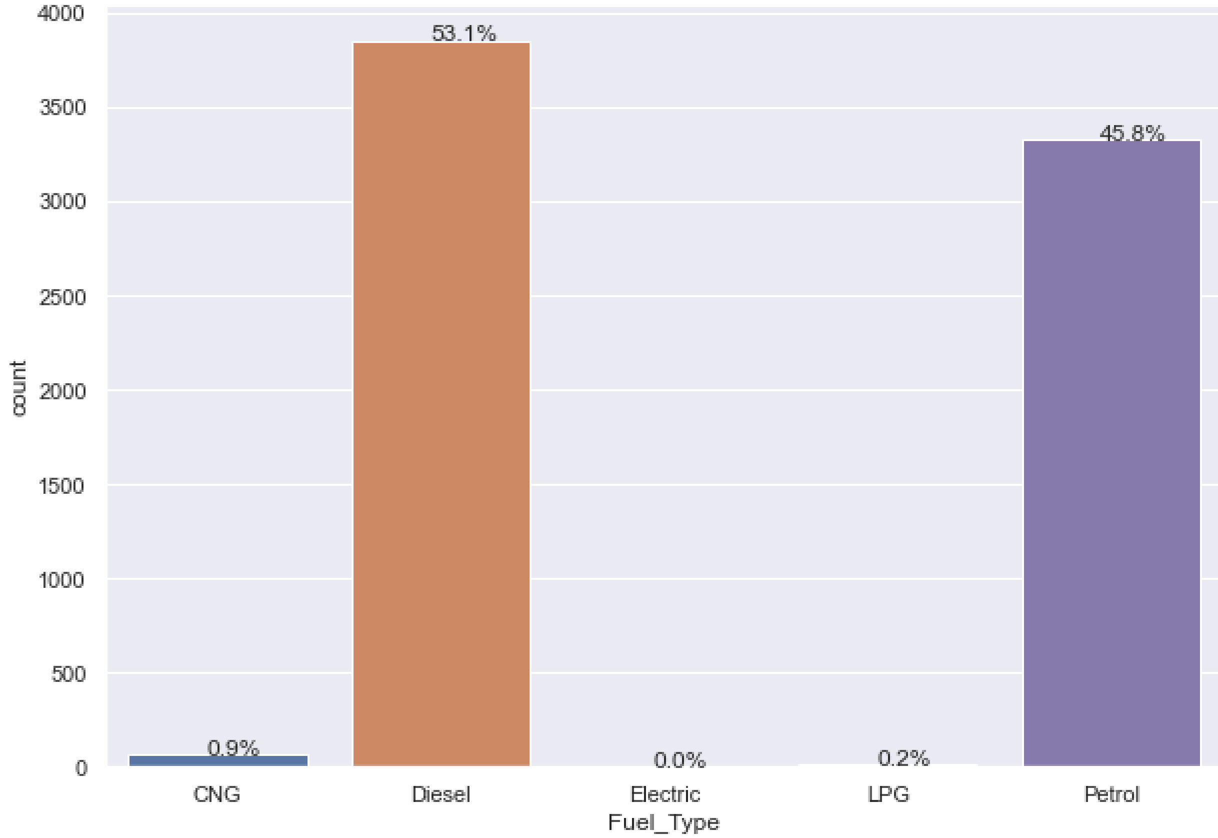
```
Top:2015
Freq:929
```



In [242…

```
bar_count_pct(df['Fuel_Type'] )
```

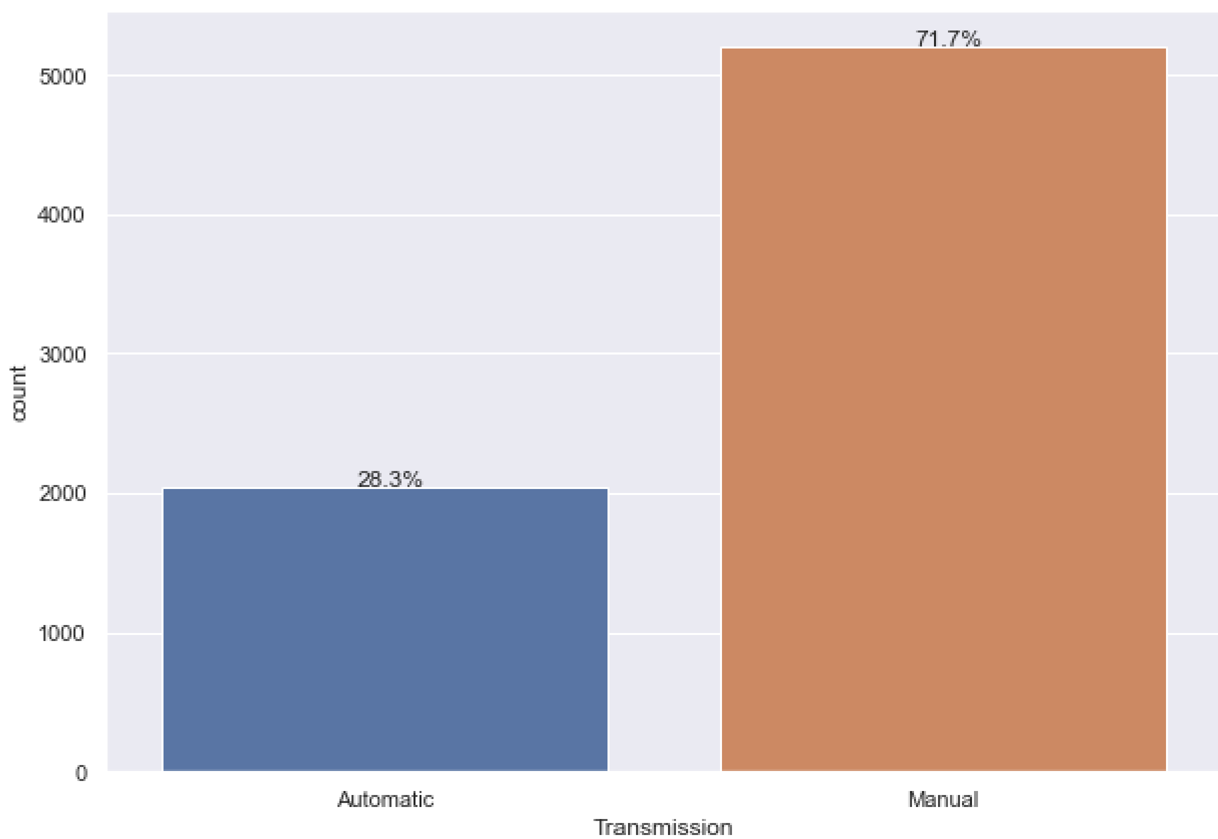
```
Top:Diesel
Freq:3852
```



In [242…

```
bar_count_pct(df['Transmission'] )
```

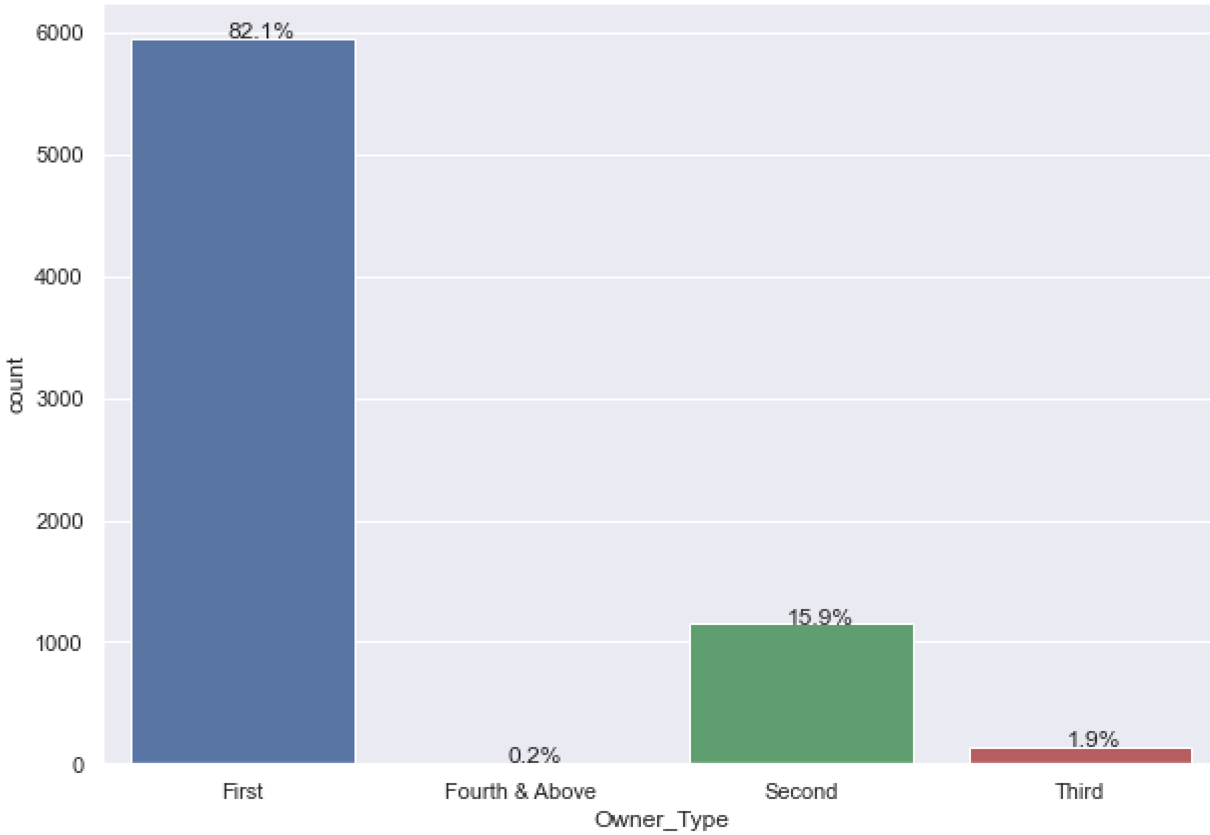
```
Top:Manual
Freq:5204
```



In [242…

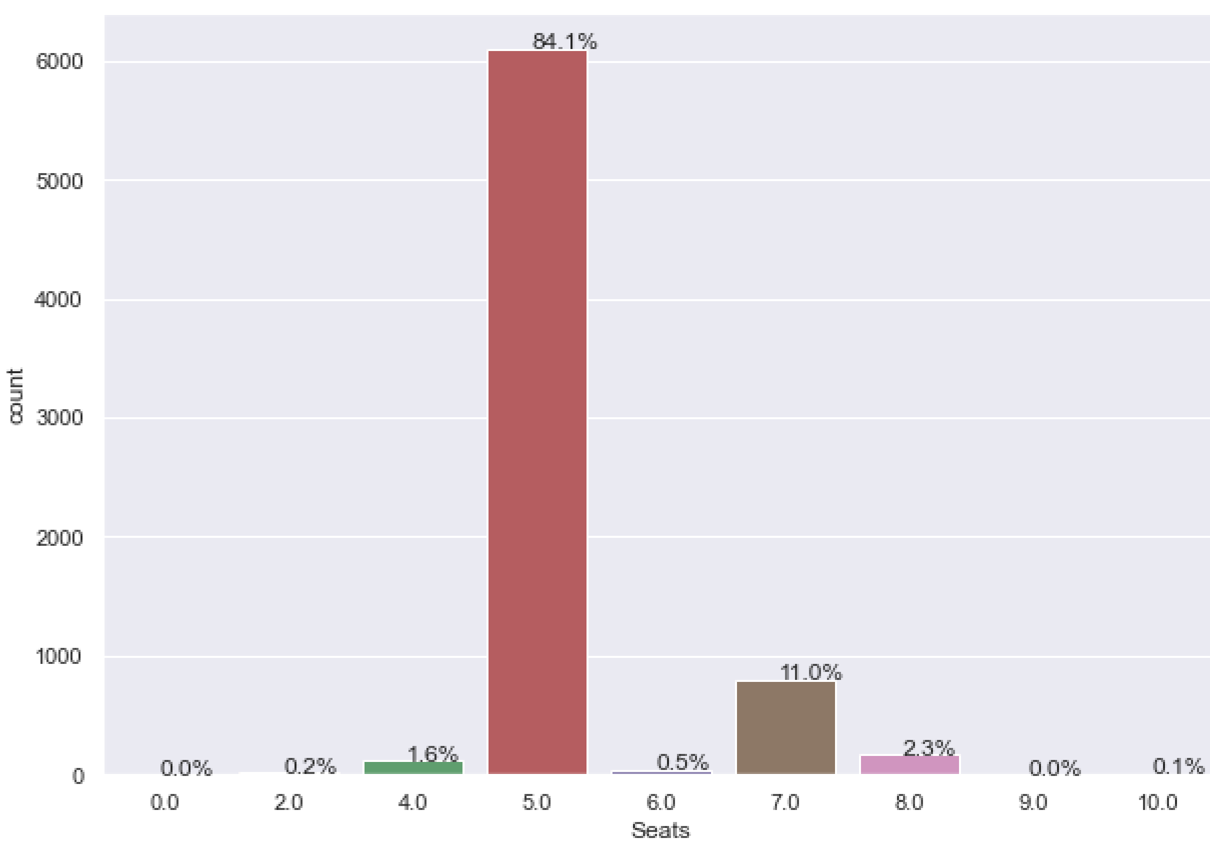
```
bar_count_pct(df['Owner_Type'] )
```

```
Top:First
Freq:5952
```



In [243…
```
bar_count_pct(df['Seats'] )
```

```
Top:5.0
Freq:6100
```

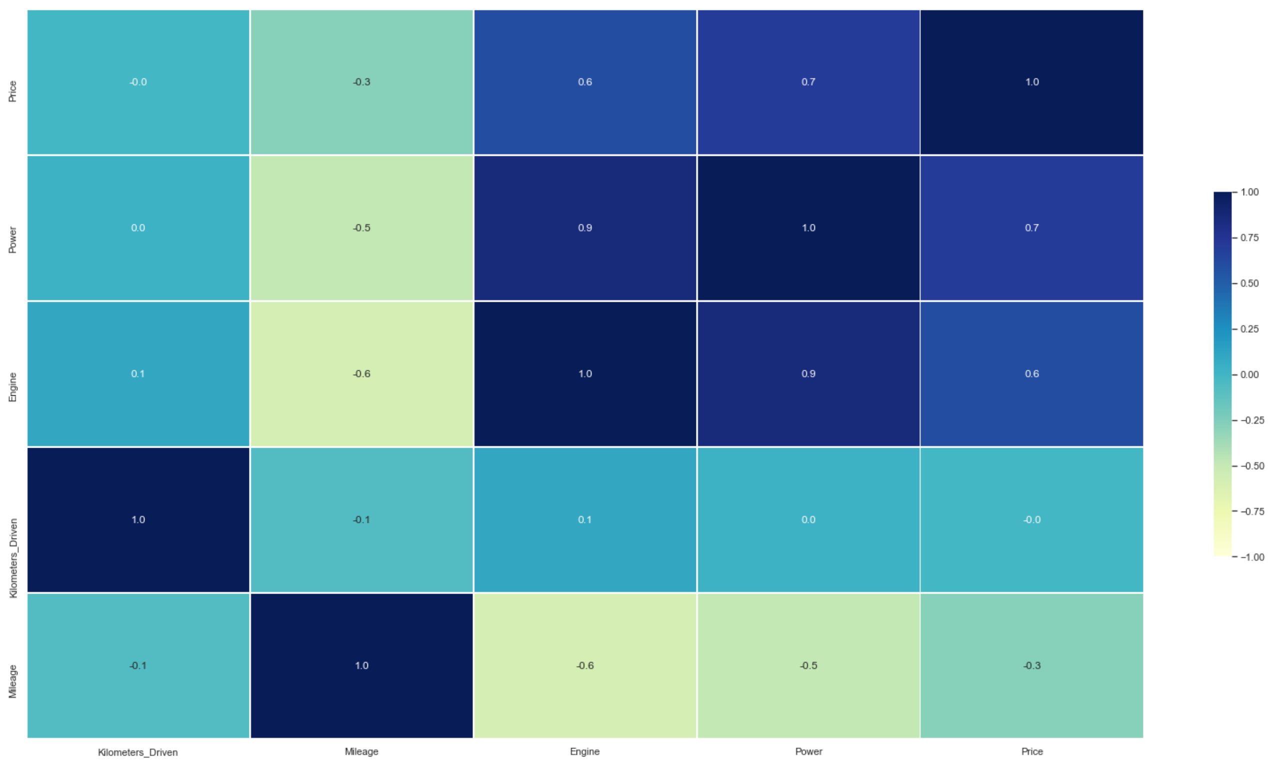


Bivaraite Analysis

Analyze Correlations

In [243...

```
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()
corr = df[numeric_columns].corr().sort_values(by=['Price'],ascending=False) # sorting c

f, ax = plt.subplots(figsize=(28, 15))

# Draw the heatmap
sns.heatmap(corr, cmap='YlGnBu', annot=True,fmt=".1f",vmin=-1, vmax=1, center= 0
            , square=False, linewidths=.7, cbar_kws={"shrink": .5});
```



Observations

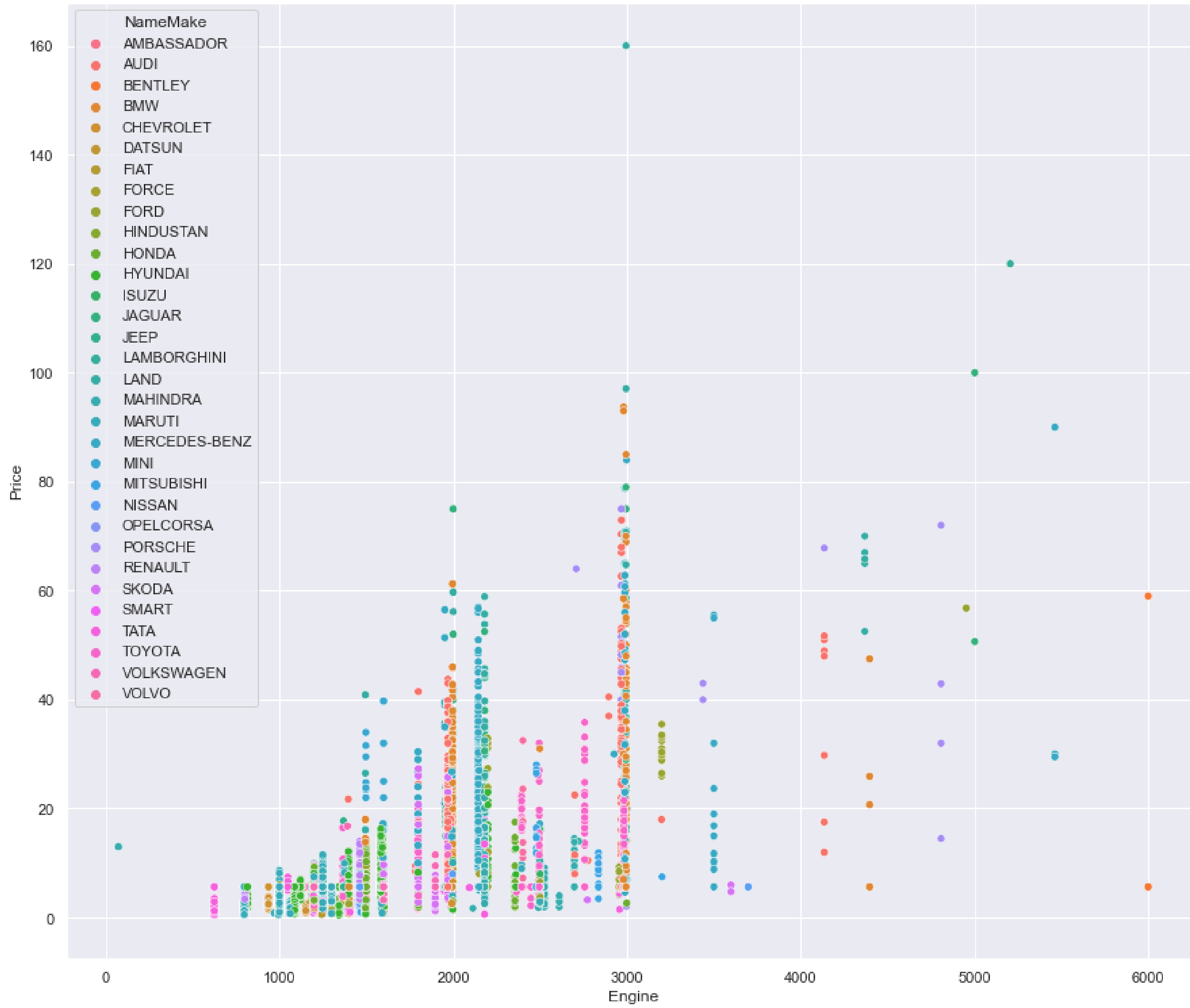
- Price is highly positively correlated with Power and Engine which means that the bigger the engine and power , the price of the vehicle is likely to increase
- Price is slightly negatively correlated with Mileage , which means that the more Mileage, the price is likely to decrease and vice-versa.

Variables that are highly correlated with Price

Price vs Engine vs NameMake

In [243...

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='NameMake', data=df);
```



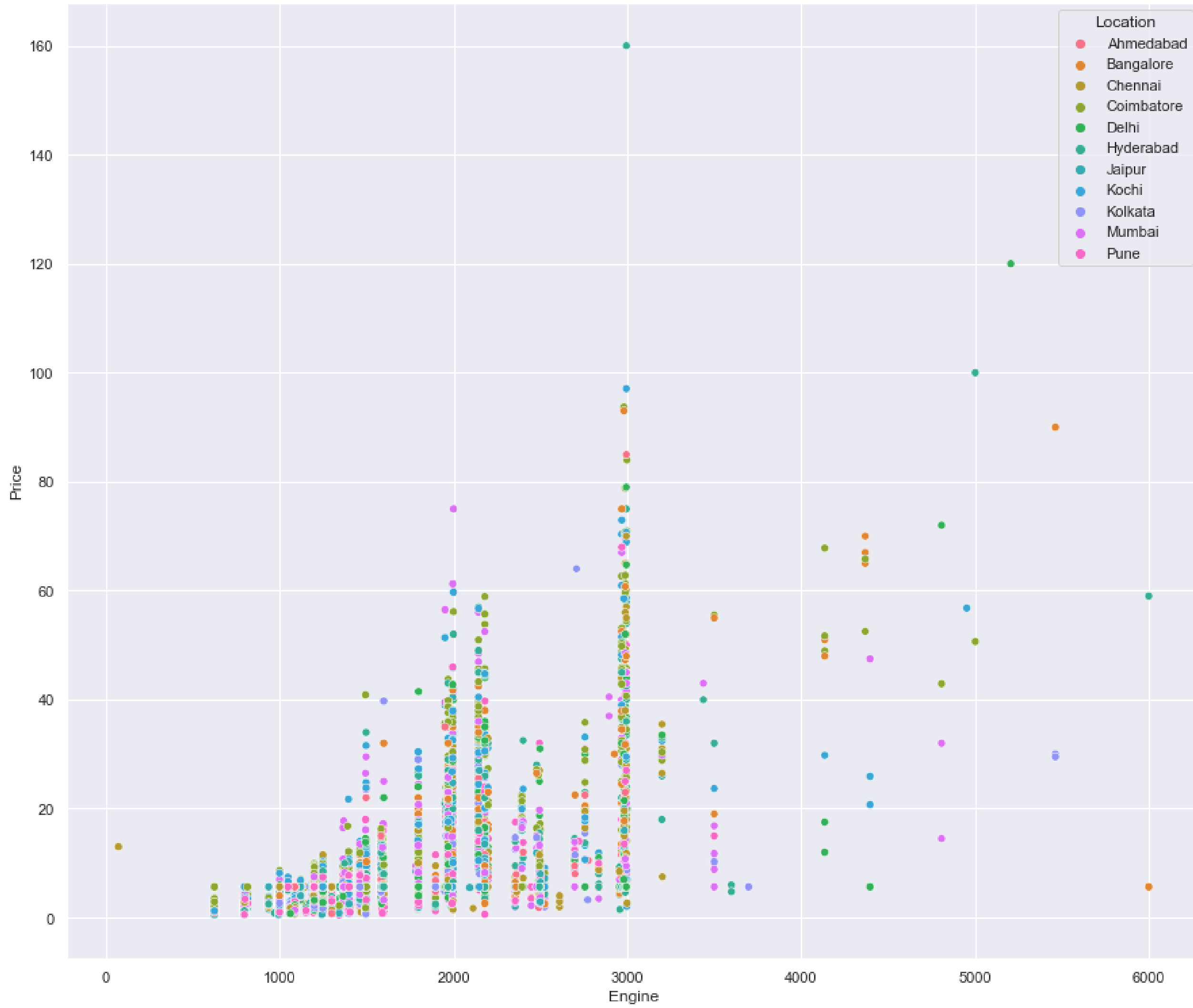
Observations

- Similare to Power variable
- Maruti , Hyundai and Honda are vehicles with smaller Engine and the Price is low as well
- Mercedes-Benz, BMW are vehicles with larger Engine and the Price is higher as well

Price vs Engine vs Location

In [243…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='Location', data=df);
```



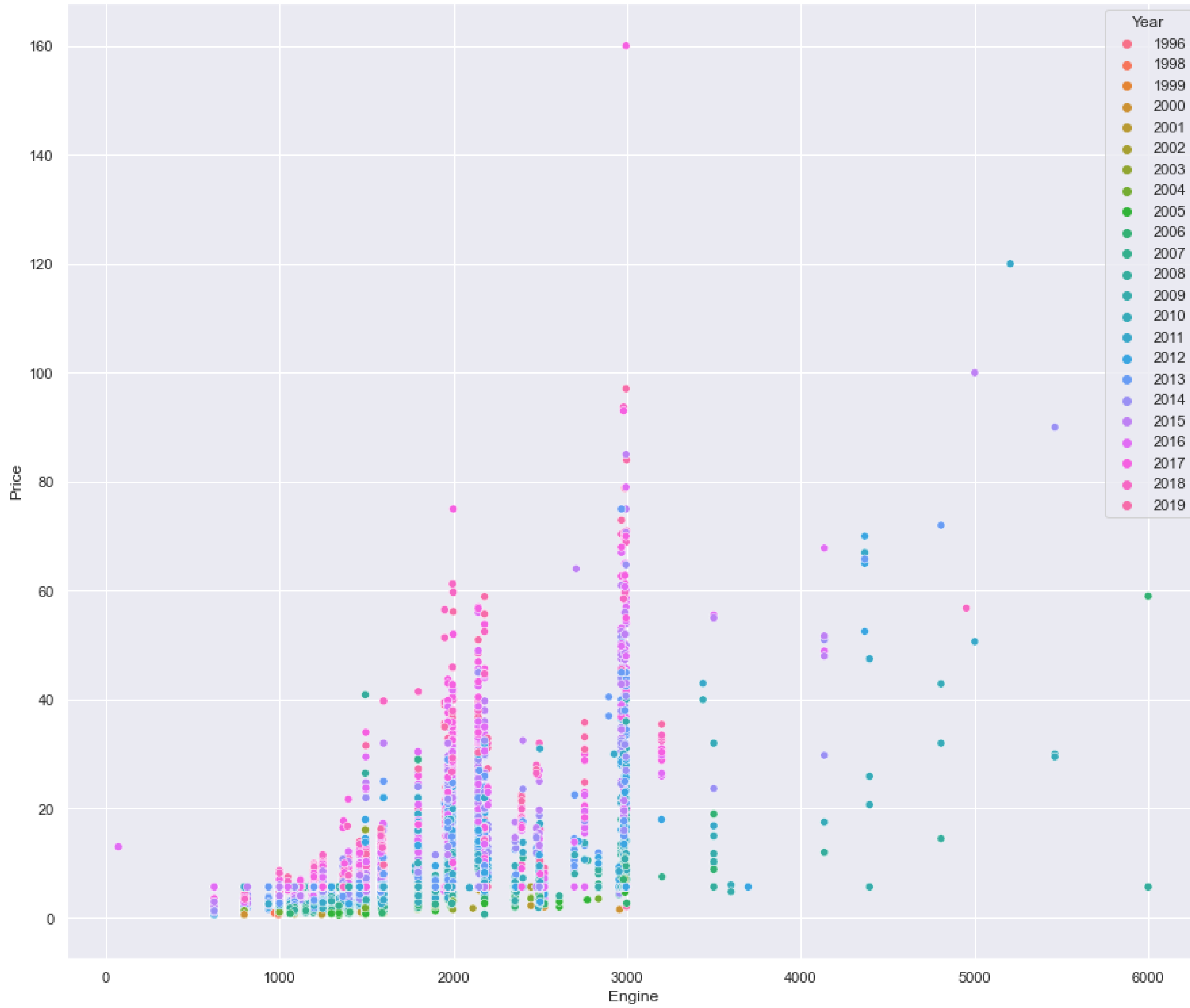
Observations

- Ther is no clear relationship between Price , Engine and Location

Price vs Engine vs Year

In [243...

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='Year', data=df);
```



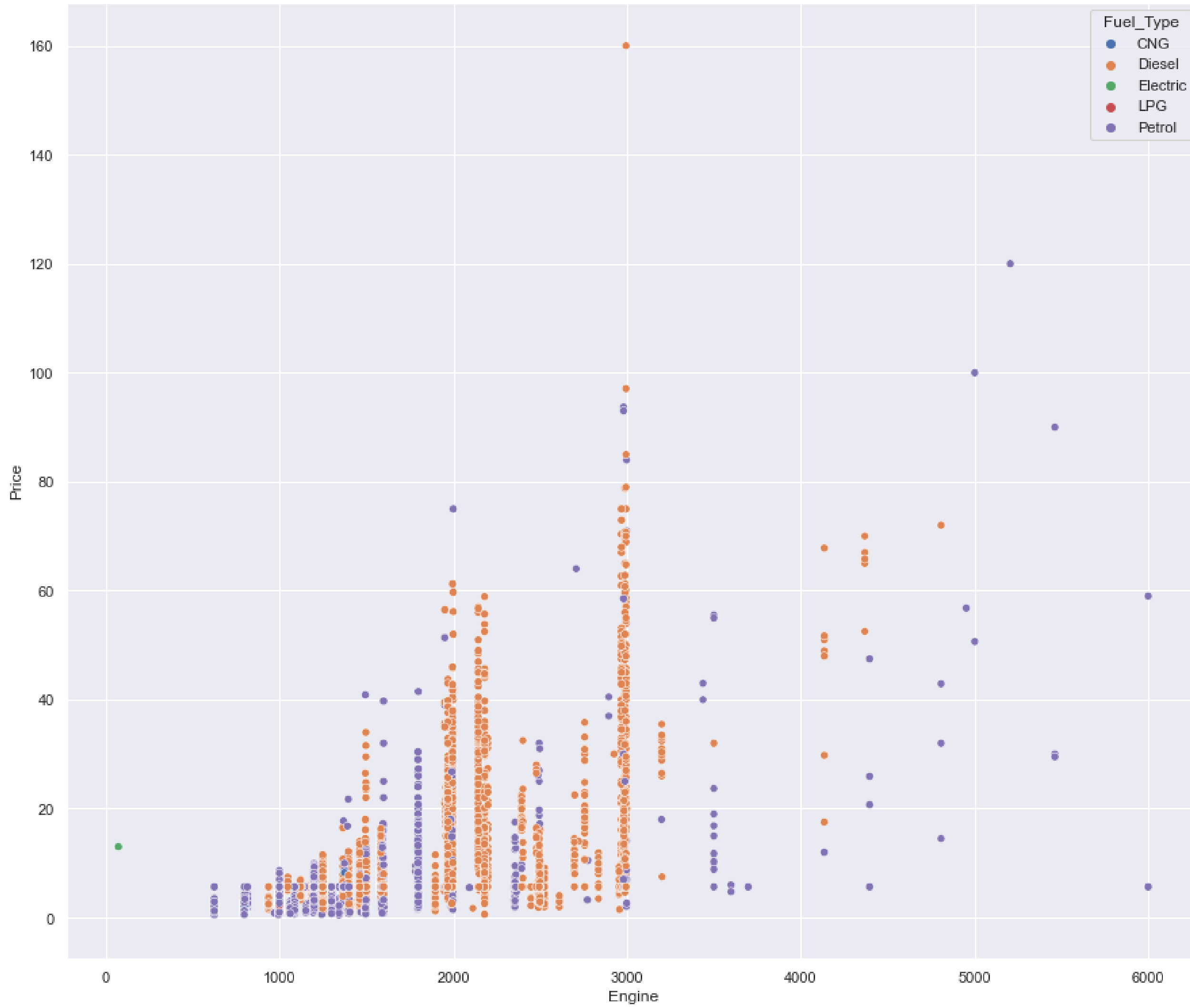
Observations

- Vehicle Year havs a direct relationship with Price regardless of Engine size.

Price vs Engine vs Fuel_Type

In [243…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='Fuel_Type', data=df);
```



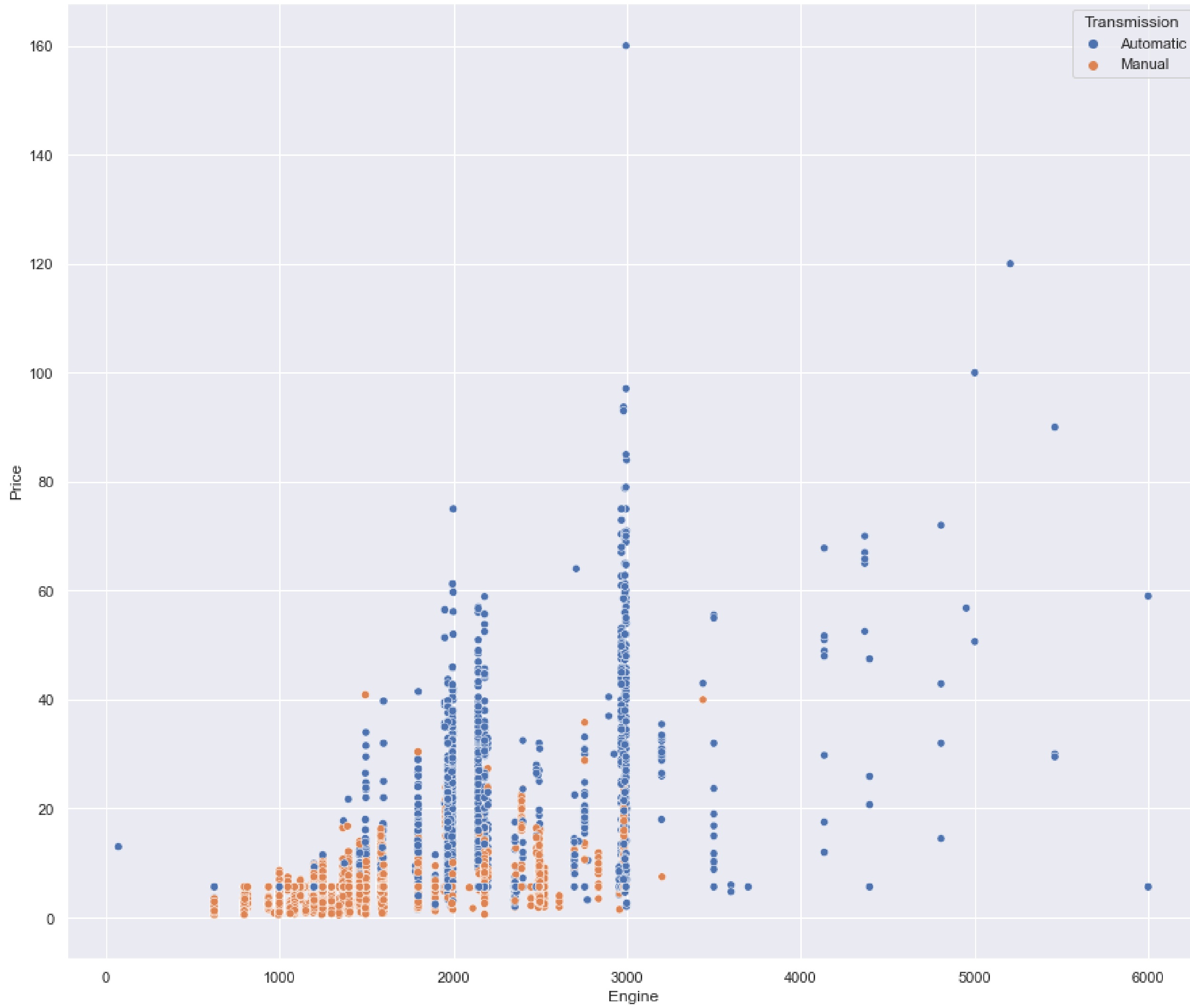
Observations

- Deisel vehicles's Price is higher between the 2000CC and 3000CC Engine size.
- Petrol vehicles's Price are more popular in the smaller Engine vehicles.

Price vs Engine vs Transmission

In [243…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='Transmission', data=df);
```



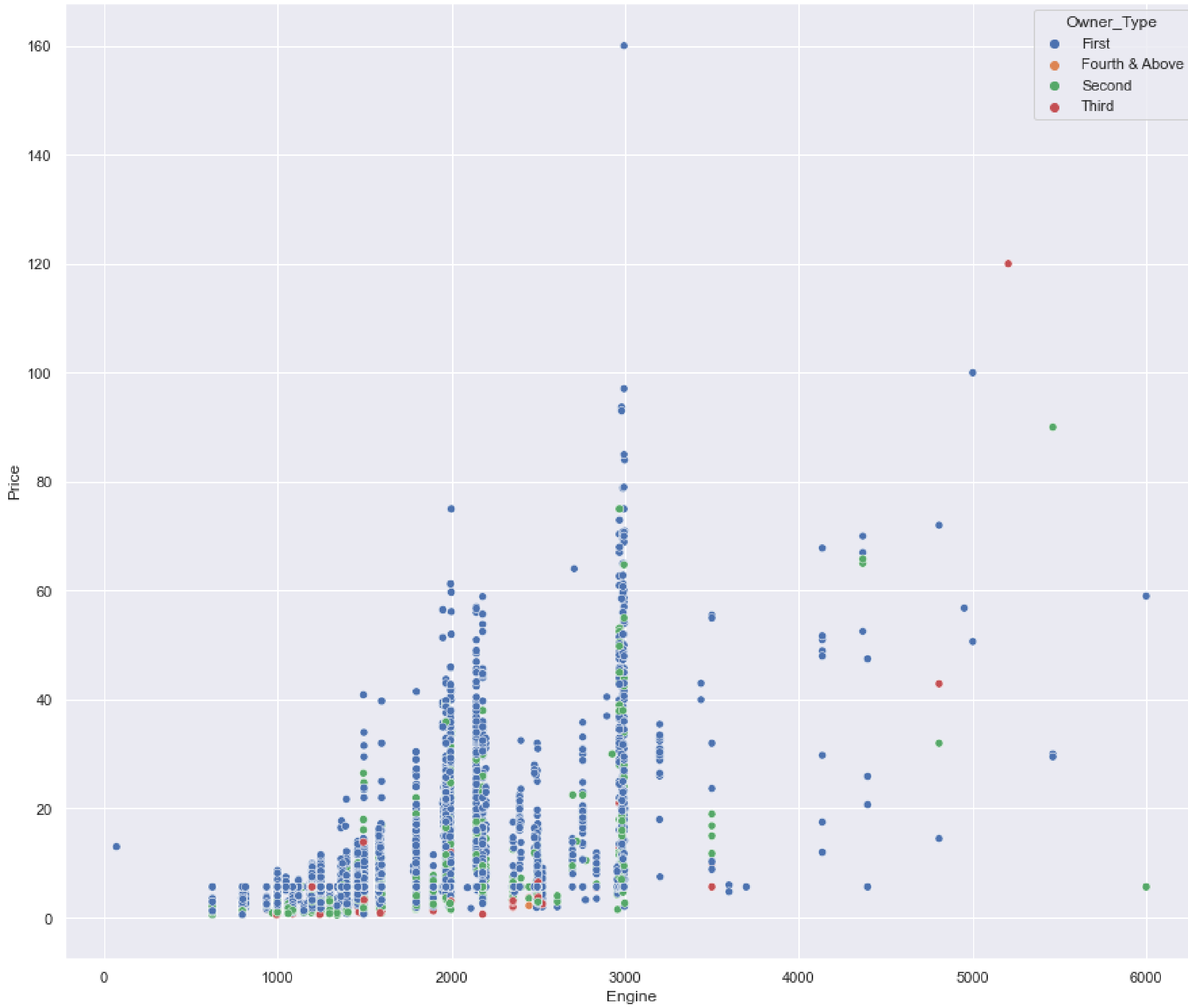
Observations

- Automatic vehicles's Price is higher between the 2000CC and 3000CC Engine size.
- Manual vehicles's Price are more popular in the smaller Engine vehicles.

Price vs Engine vs Owner_Type

In [243…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='Owner_Type', data=df);
```



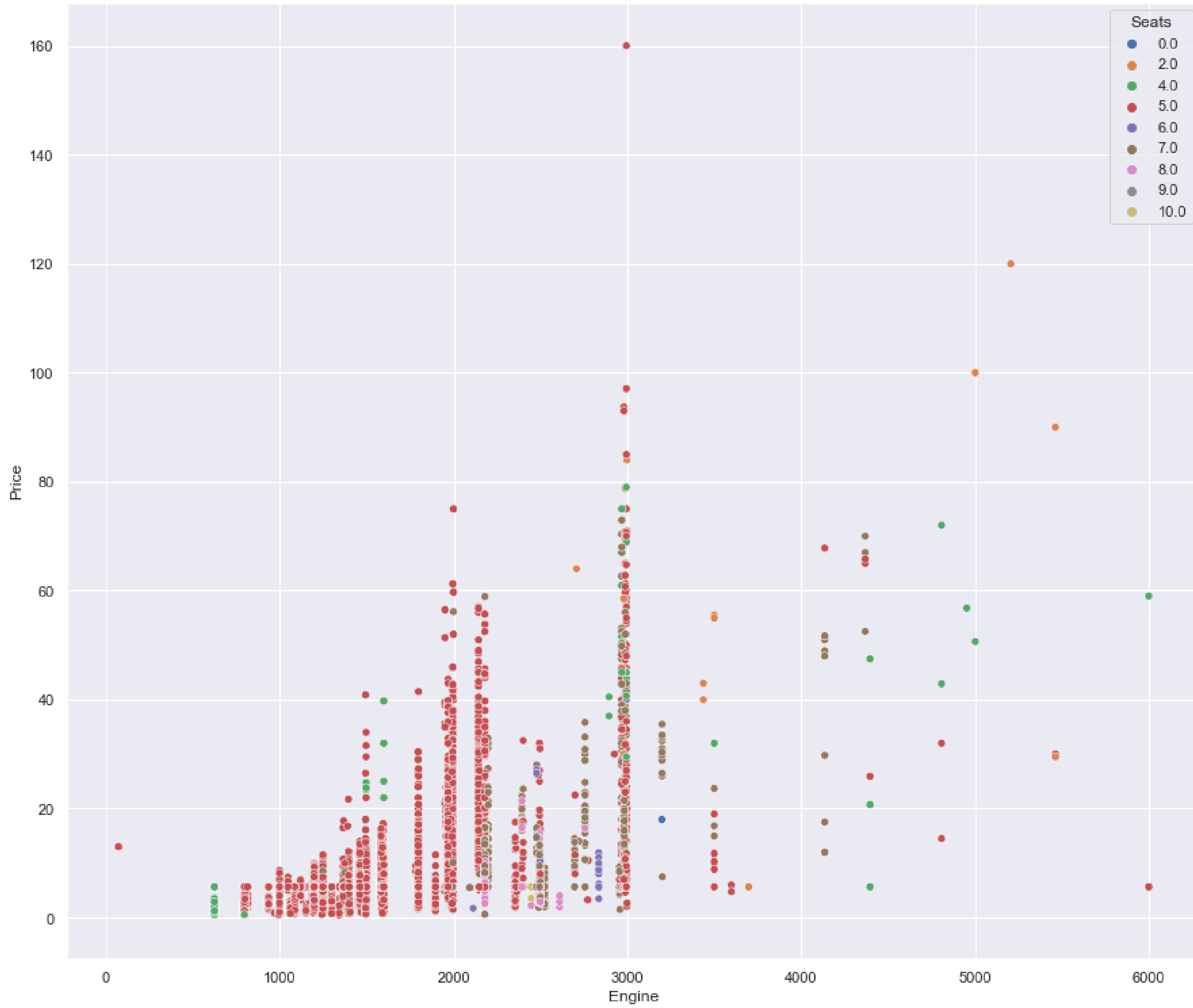
Observations

- First Owner_types' Price is higher than any other Owner Type.

Price vs Engine vs Seats

In [243…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Engine', hue='Seats', data=df);
```



Observations

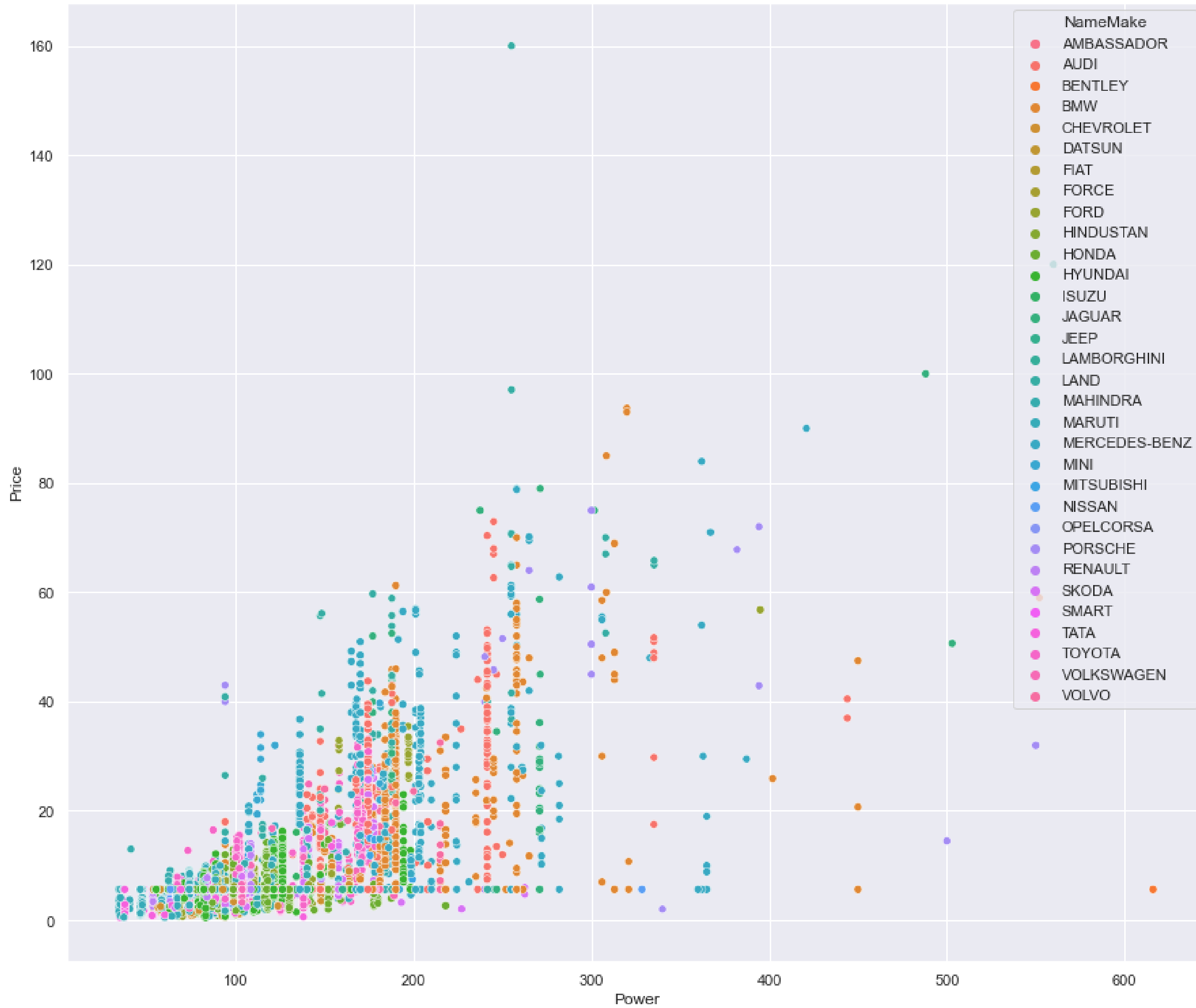
- 5-Seat vehicles's are the more popular choice and for lower size Engines, the Price seems to be high as well.

In [ ]:

Price vs Power vs NameMake

In [243…

```
# Lets look visualize the relationship
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='NameMake', data=df);
```



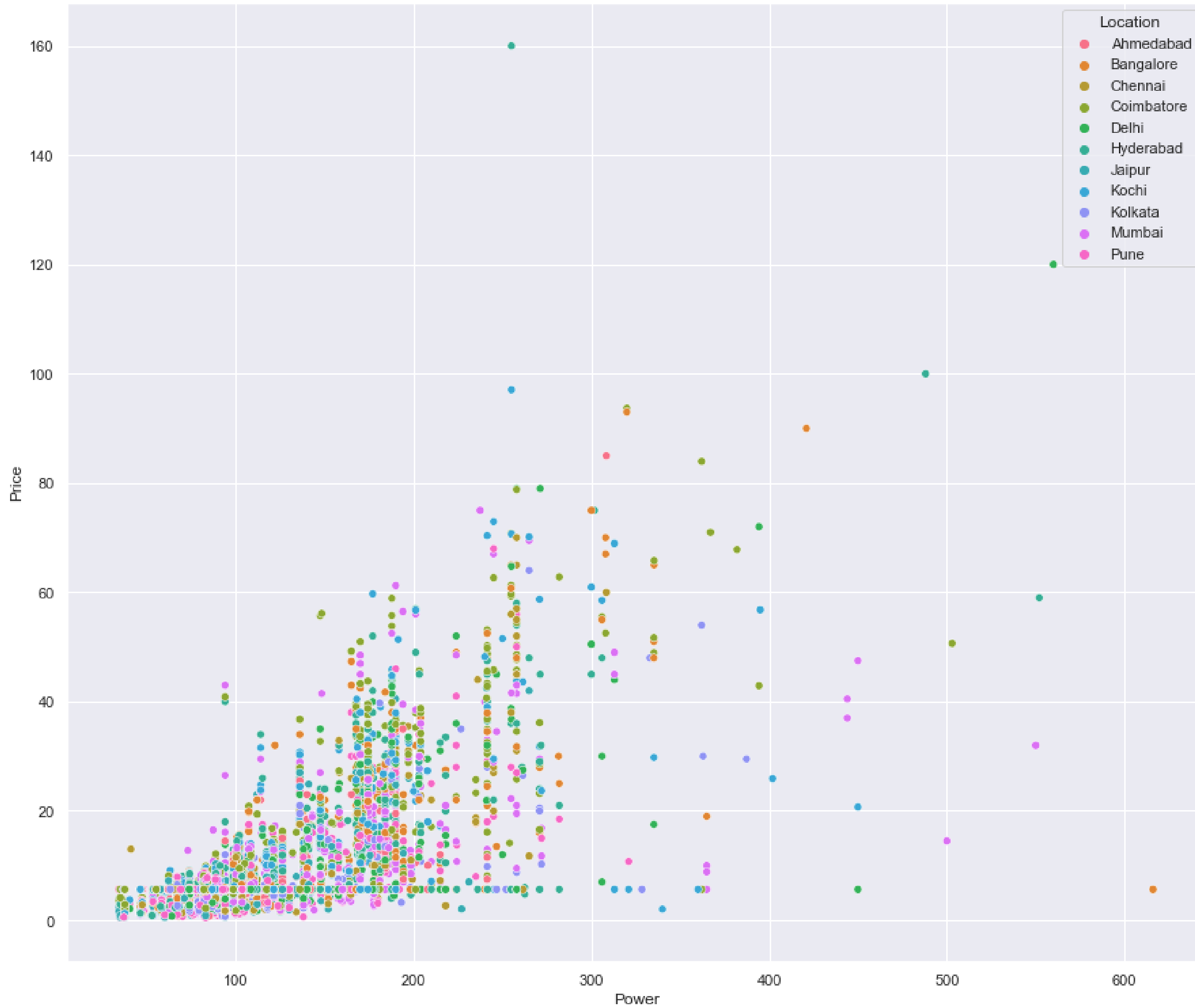
Observations

- Maruti , Hyundai and Honda are vehicles with low Power and the Price is low as well
- Mercedes-Benz, BMW are vehicles with higher Power and the Price is higher as well

Price vs Power vs Location

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='Location', data=df);
```



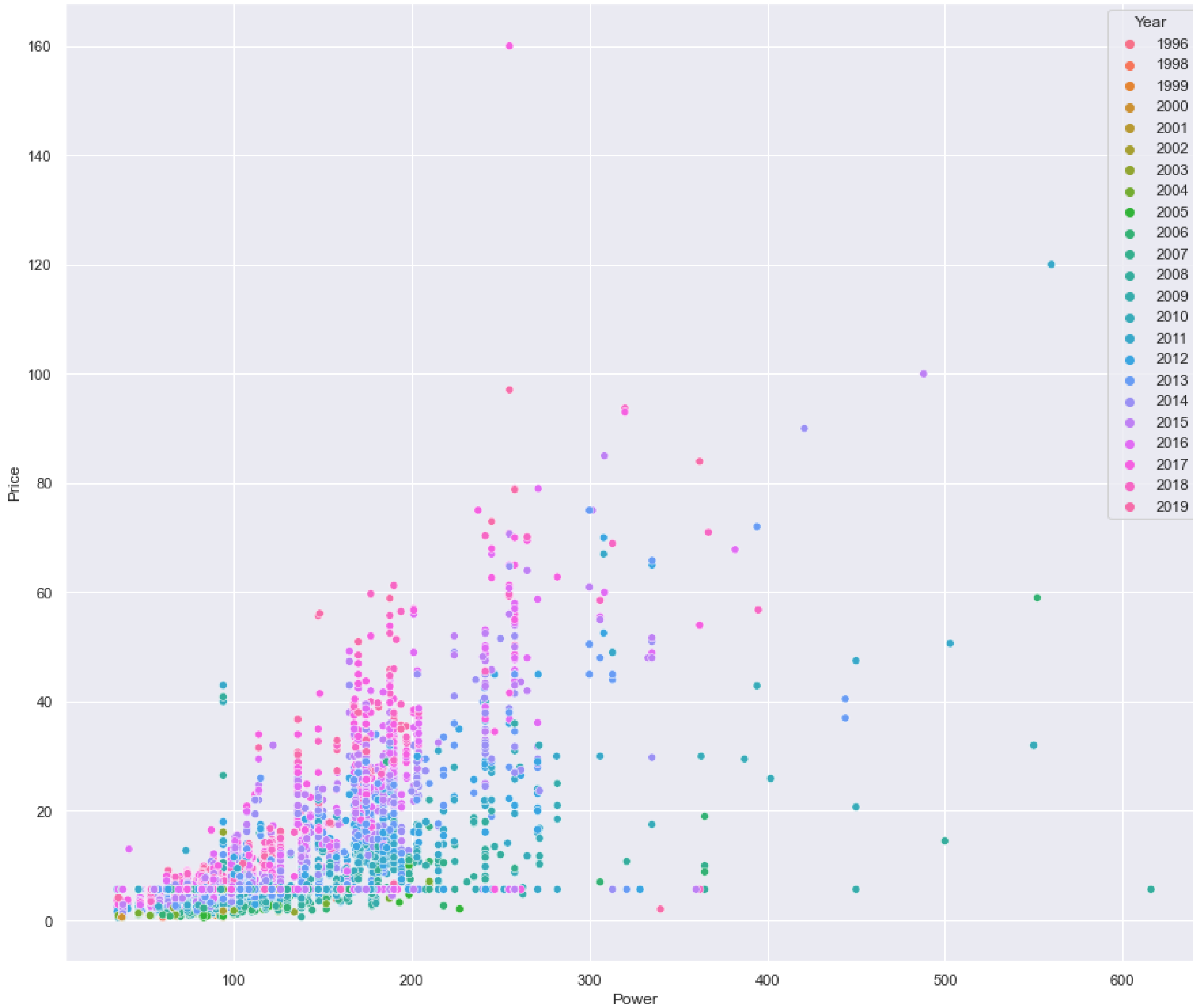
Observations

- Ther is no clear relationship between Price , Power and Location.

Price vs Power vs Year

In [244...

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='Year', data=df);
```



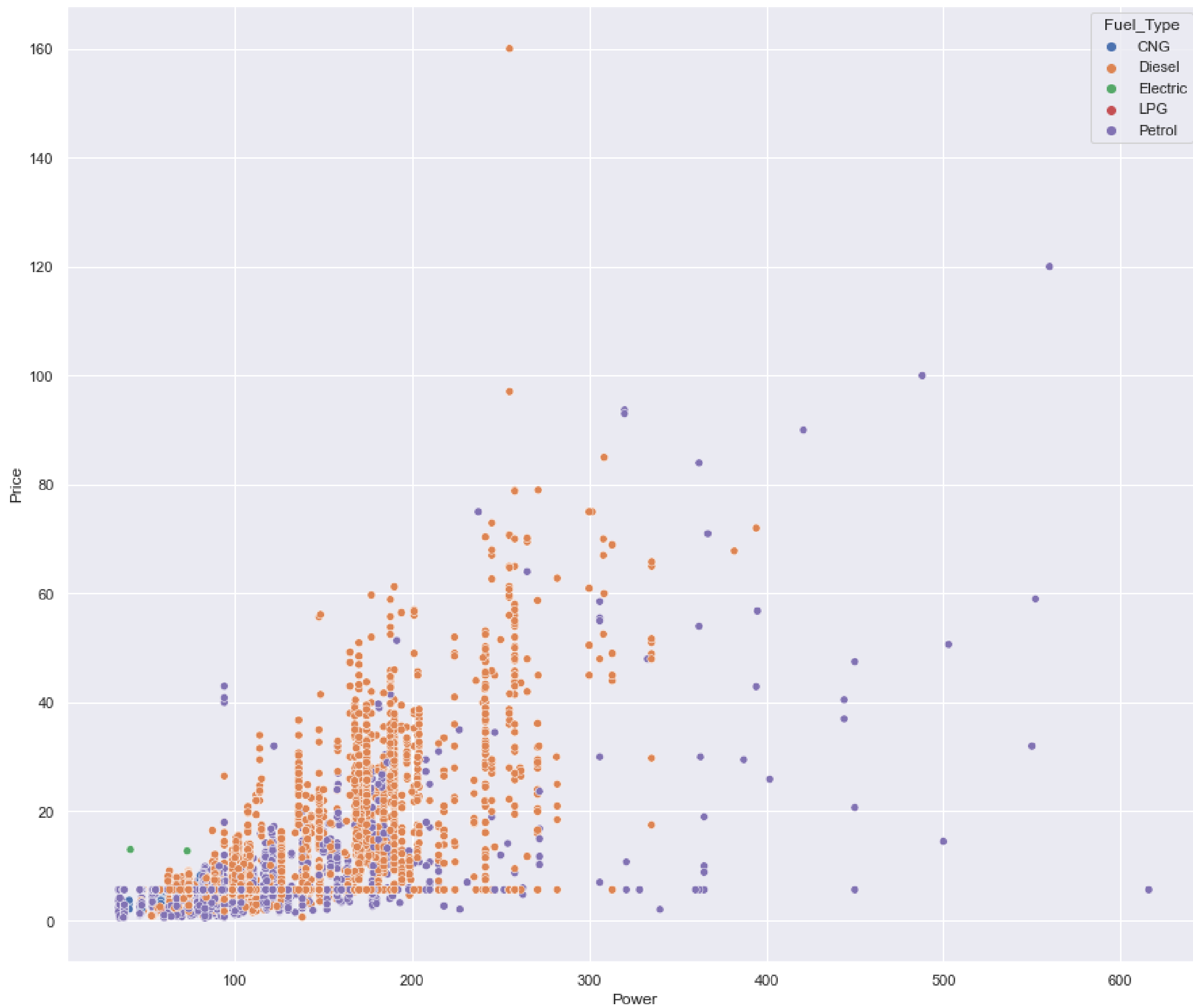
Observations

- Vehicle's Year is directly related to the Price, regardless of Power

Price vs Power vs Fuel_Type

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='Fuel_Type', data=df);
```



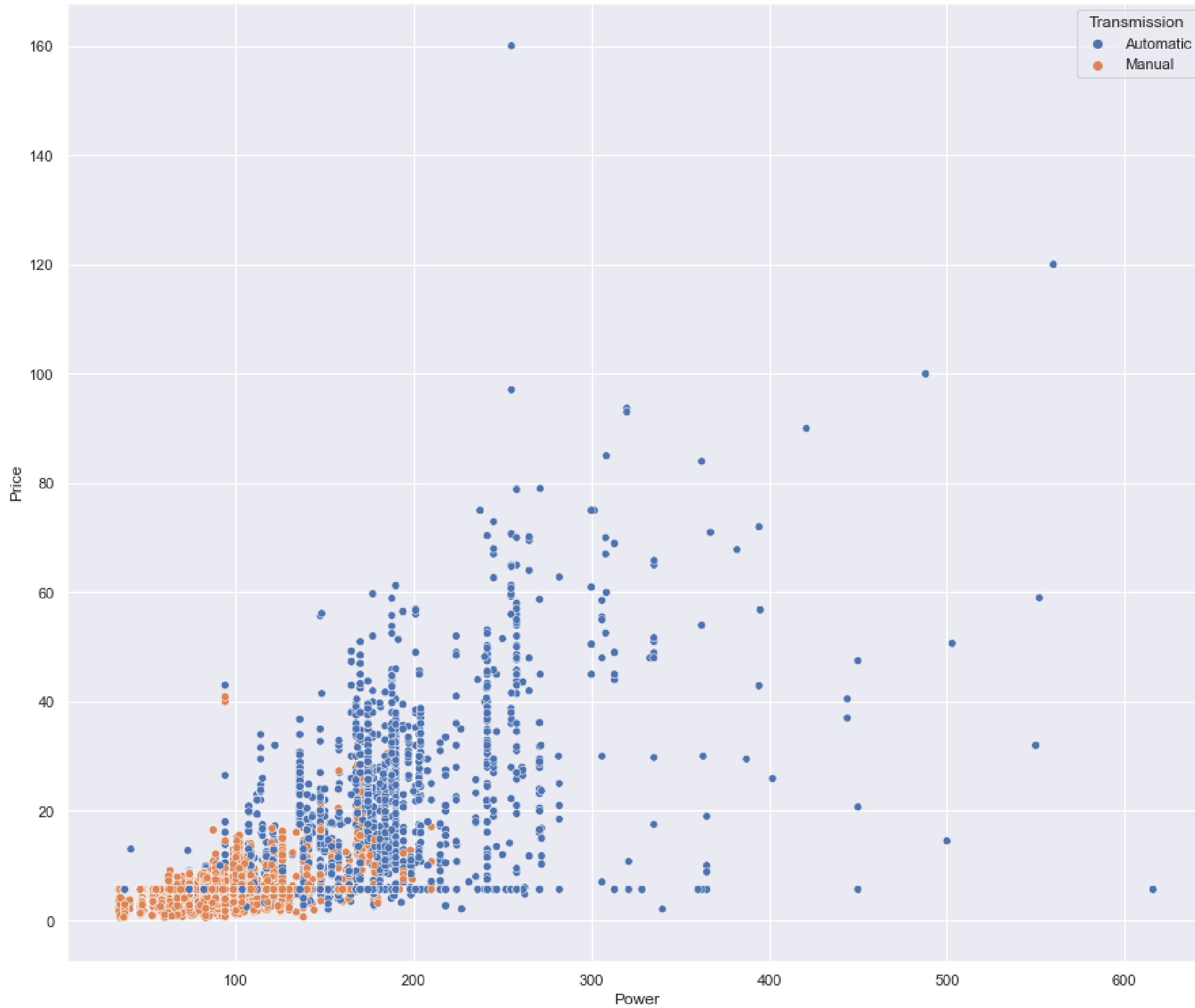
Observations

- Deisel cars' Price is higher than any other type regardless of Power

Price vs Power vs Transmission

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='Transmission', data=df);
```

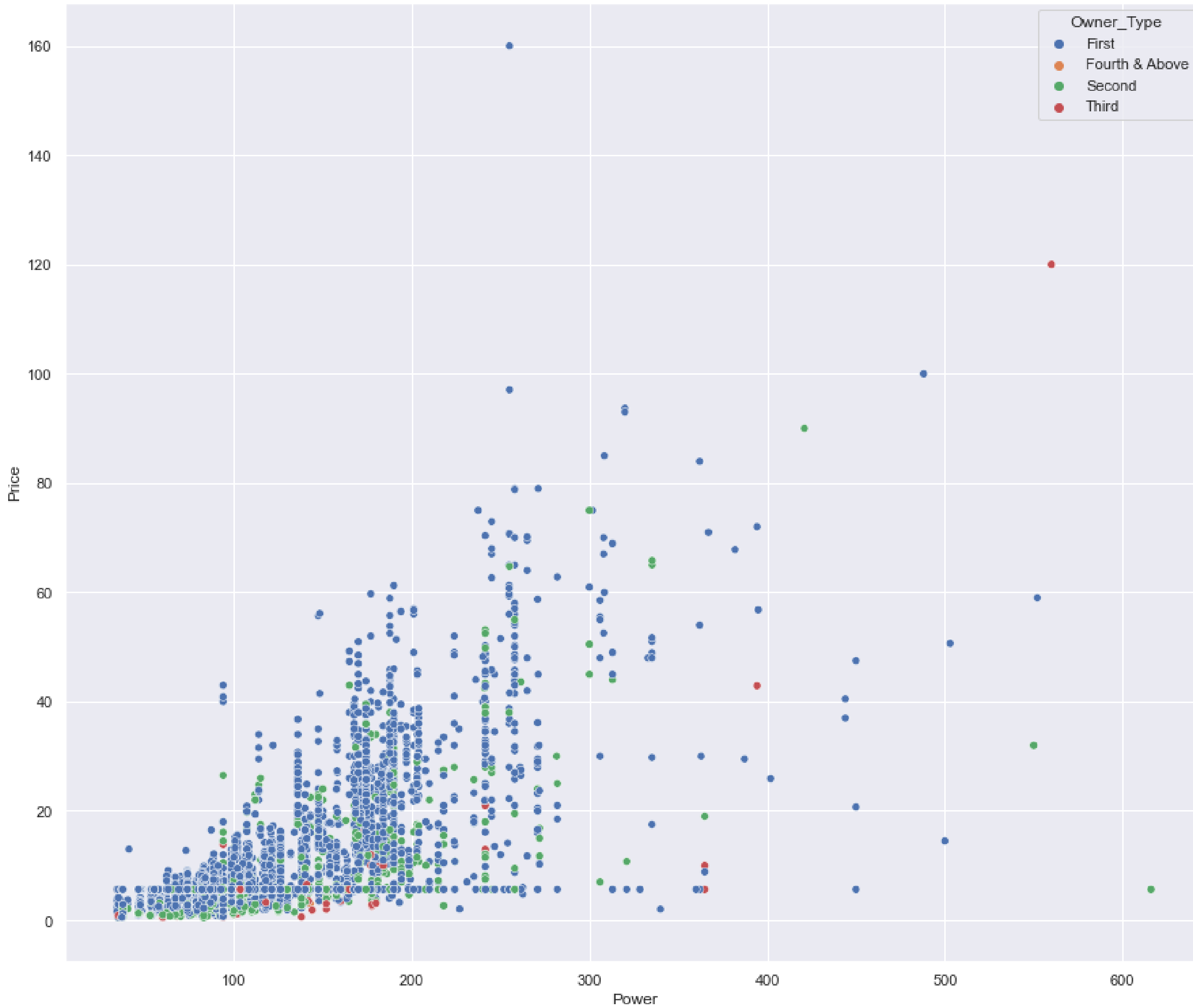


Observations

- Automatic cars Price is higher than Manual regardless of Power

Price vs Power vs Owner_Type

In [244…
```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='Owner_Type', data=df);
```



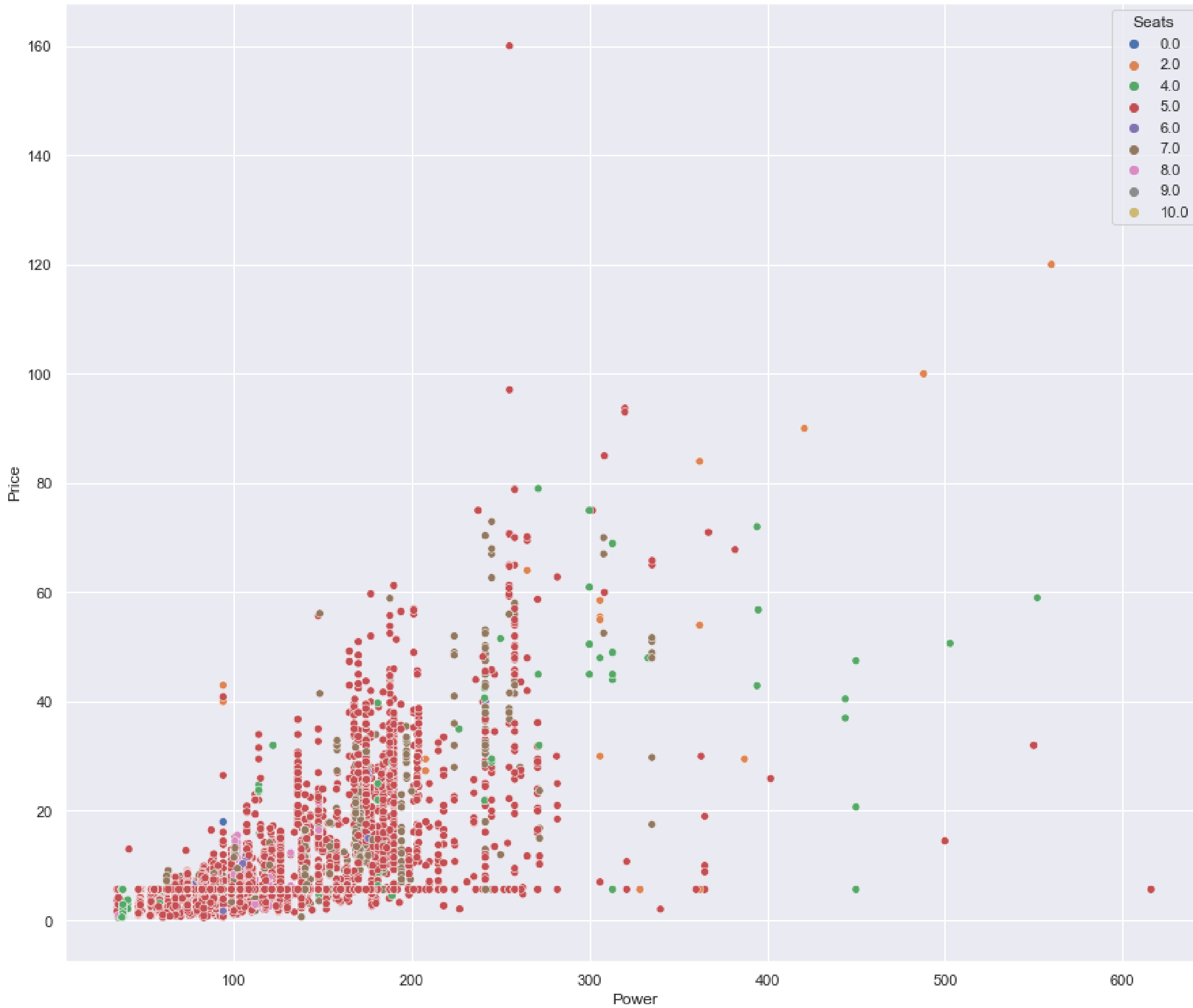
Observations

- First Owner_Type's Price is higher regardless of Power

Price vs Power vs Seats

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Power', hue='Seats', data=df);
```



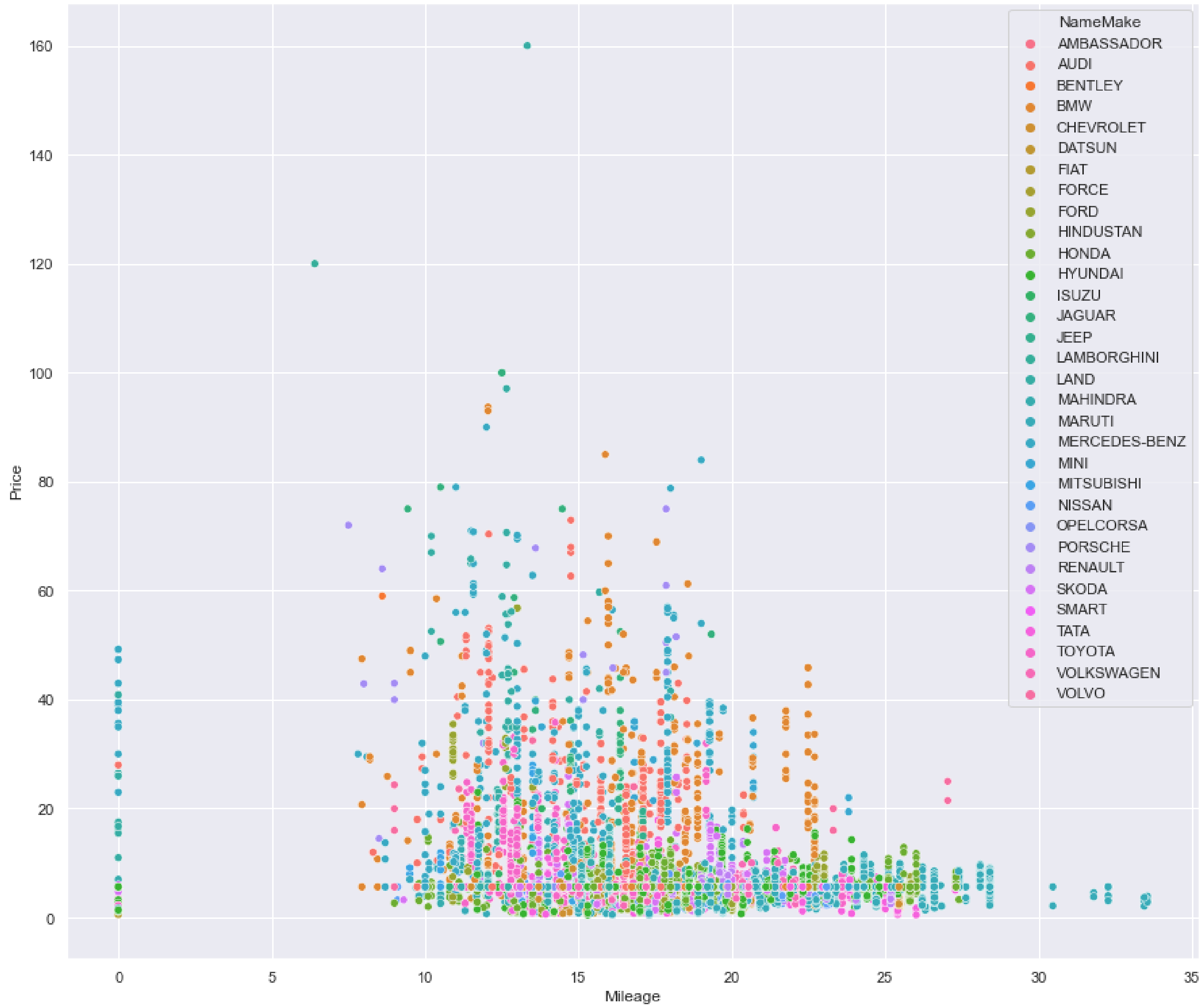
Observations

- 5-Seat cars are the most popular option,and on average the Price is higher than all the other Seat values regardless of Power

Price vs Mileage vs NameMake

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='NameMake', data=df);
```



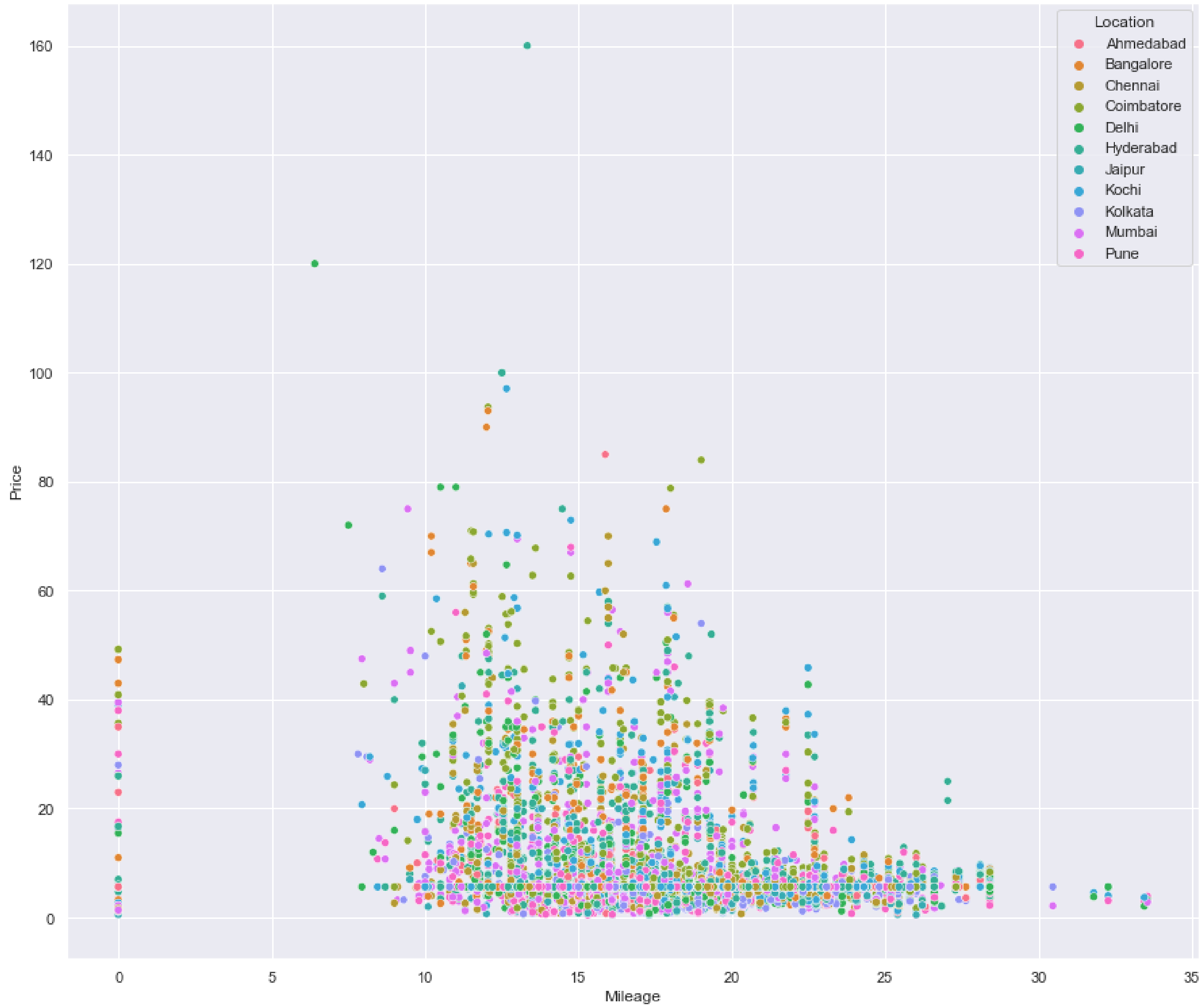
Observations

- Maruti , Hyundai and Honda are vehicles with high Mileage value and the Price is i negatively correlated.

Price vs Mileage vs Location

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='Location', data=df);
```



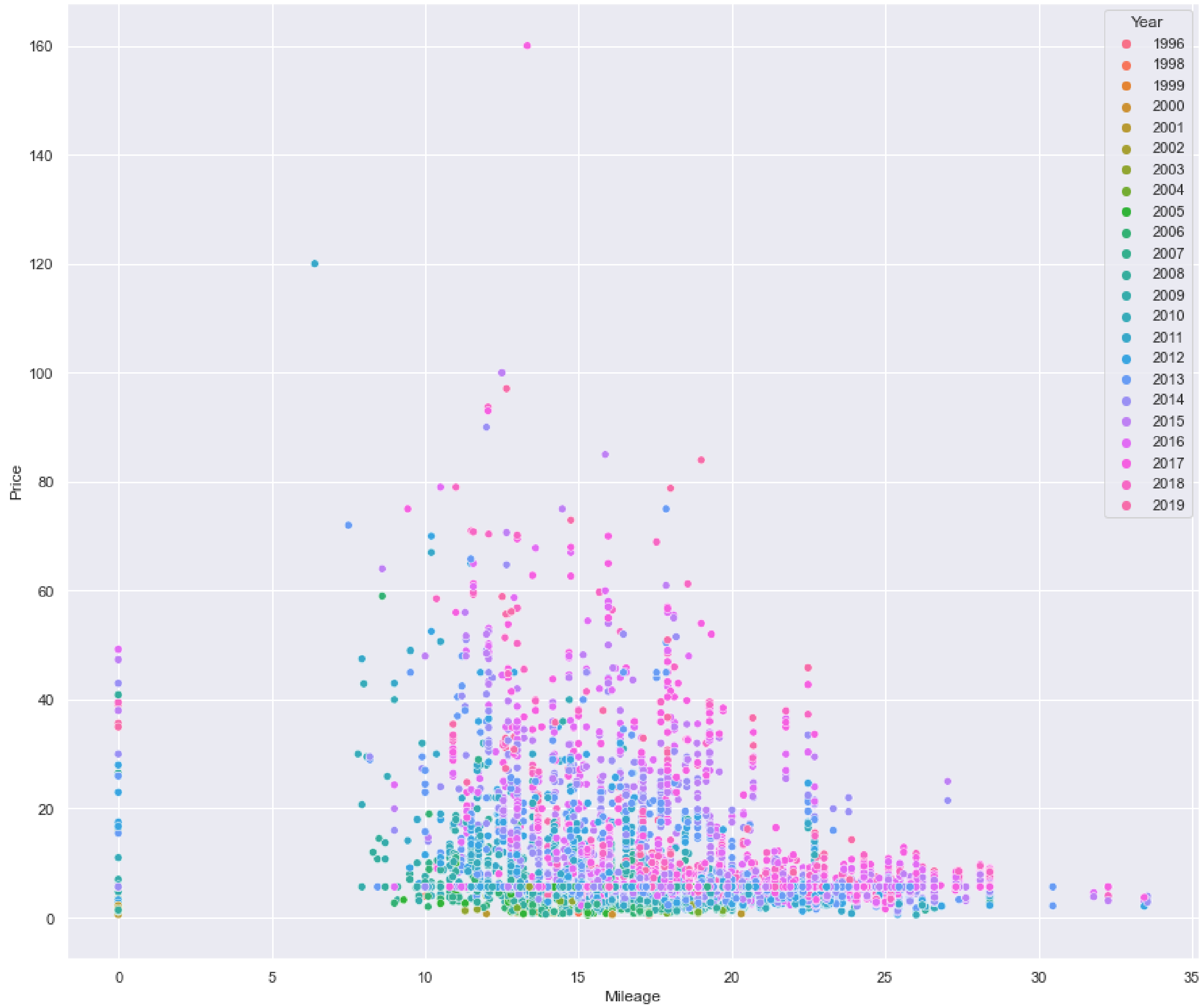
Observations

- There is no clear relationship between Price , Mileage and Location.

Price vs Mileage vs Year

In [244...

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='Year', data=df);
```



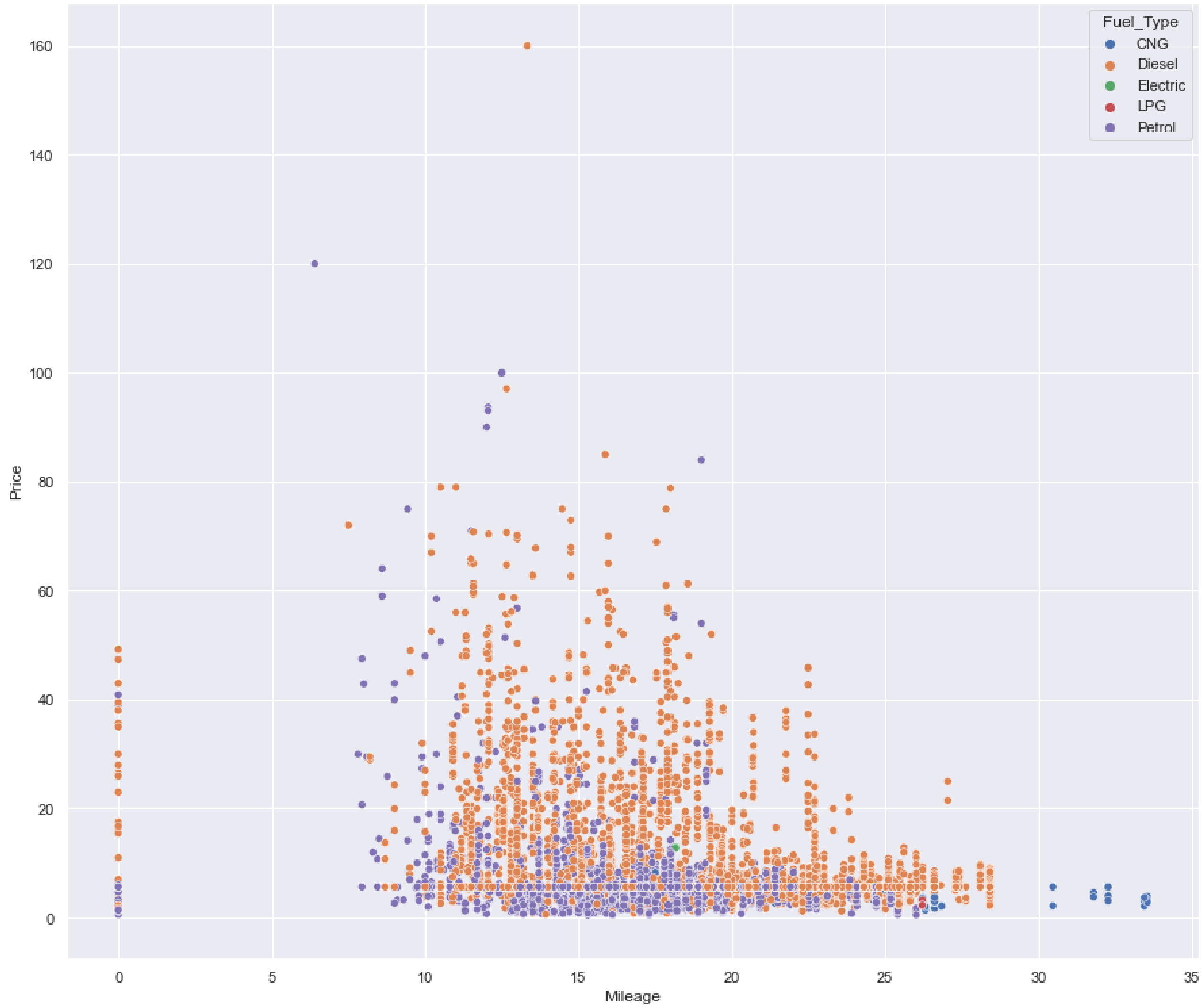
Observations

- Cars with high Mileage value seem to have lower Prices regardless of Year. There some exceptions with newer cars

Price vs Mileage vs Fuel_Type

In [244…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='Fuel_Type', data=df);
```



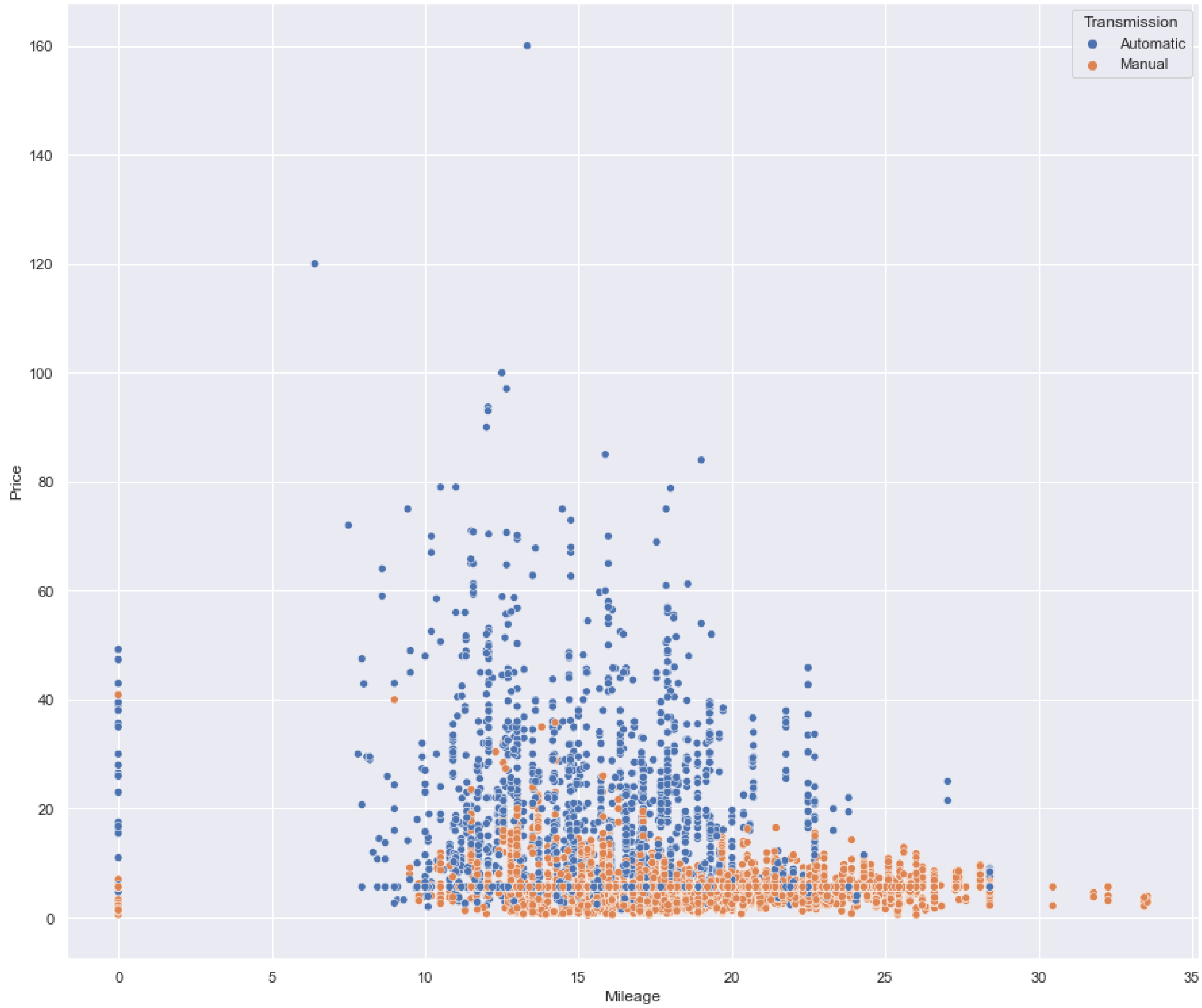
Observations

- Cars with high Mileage value seem to have lower Prices regardless of Fuel_Type. There some exceptions with Deisel types

Price vs Mileage vs Transmission

In [245...

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='Transmission', data=df);
```



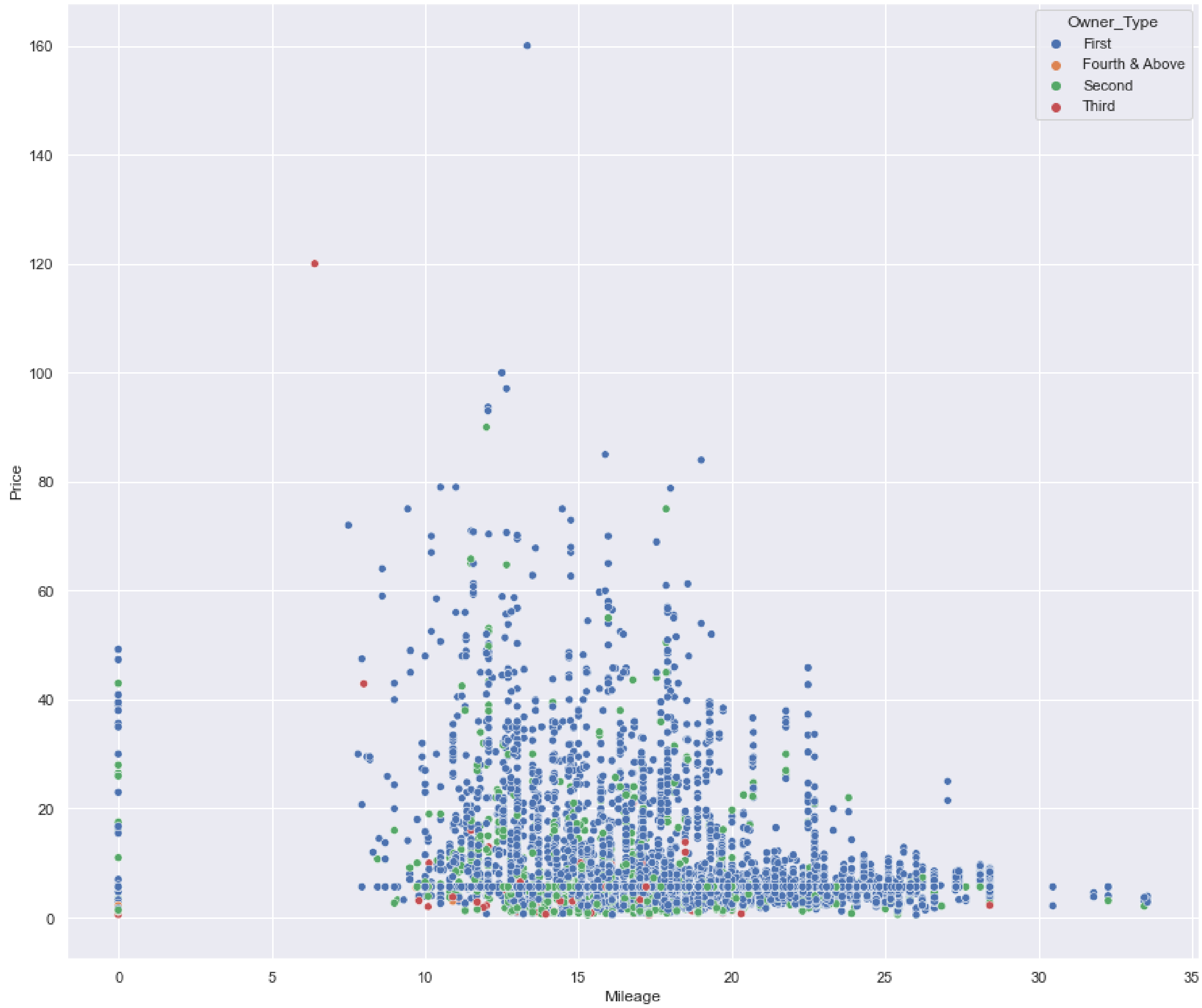
Observations

- Automatic Cars with higher Price than Manual, regardlees of Mileage.

Price vs Mileage vs Owner_Type

In [245…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='Owner_Type', data=df);
```



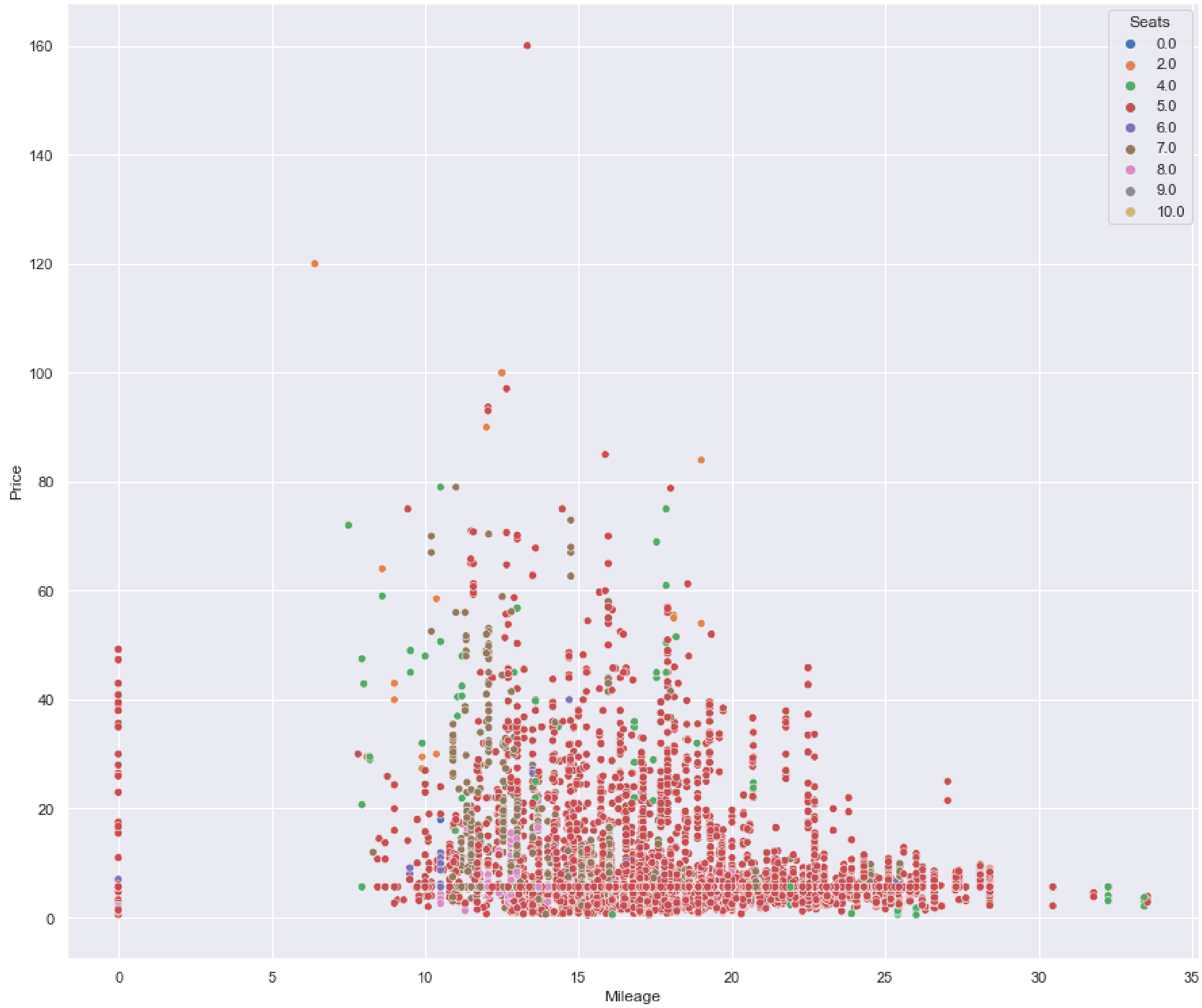
Observations

- First Owner Type Cars have higher Price than any other type, regardlees of Mileage.

Price vs Mileage vs Seats

In [245…

```
plt.figure(figsize=(15,13))
sns.scatterplot(y='Price', x='Mileage', hue='Seats', data=df);
```



Observations

- 5-Seat Cars are the most popular option.

In [245…

```
# we will replace missing values in every column with its medain
medianFiller = lambda x: x.fillna(x.median())
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()
df[numeric_columns] = df[numeric_columns].apply(medianFiller,axis=0)
#df["Seats"] = df["Seats"].apply(medianFiller,axis=0)
```

In [245…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               7253 non-null   category
 1   Location           7253 non-null   category
 2   Year               7253 non-null   category
 3   Kilometers_Driven  7253 non-null   int64
 4   Fuel_Type          7253 non-null   category
 5   Transmission       7253 non-null   category
 6   Owner_Type         7253 non-null   category
 7   Mileage            7253 non-null   float64
 8   Engine             7253 non-null   float64
 9   Power              7253 non-null   float64
```

```
 10  Seats              7253 non-null   category
 11  Price              7253 non-null   float64
 12  NameMake           7253 non-null   category
dtypes: category(8), float64(4), int64(1)
memory usage: 446.7 KB
```

Outliers Treatment

In [245...

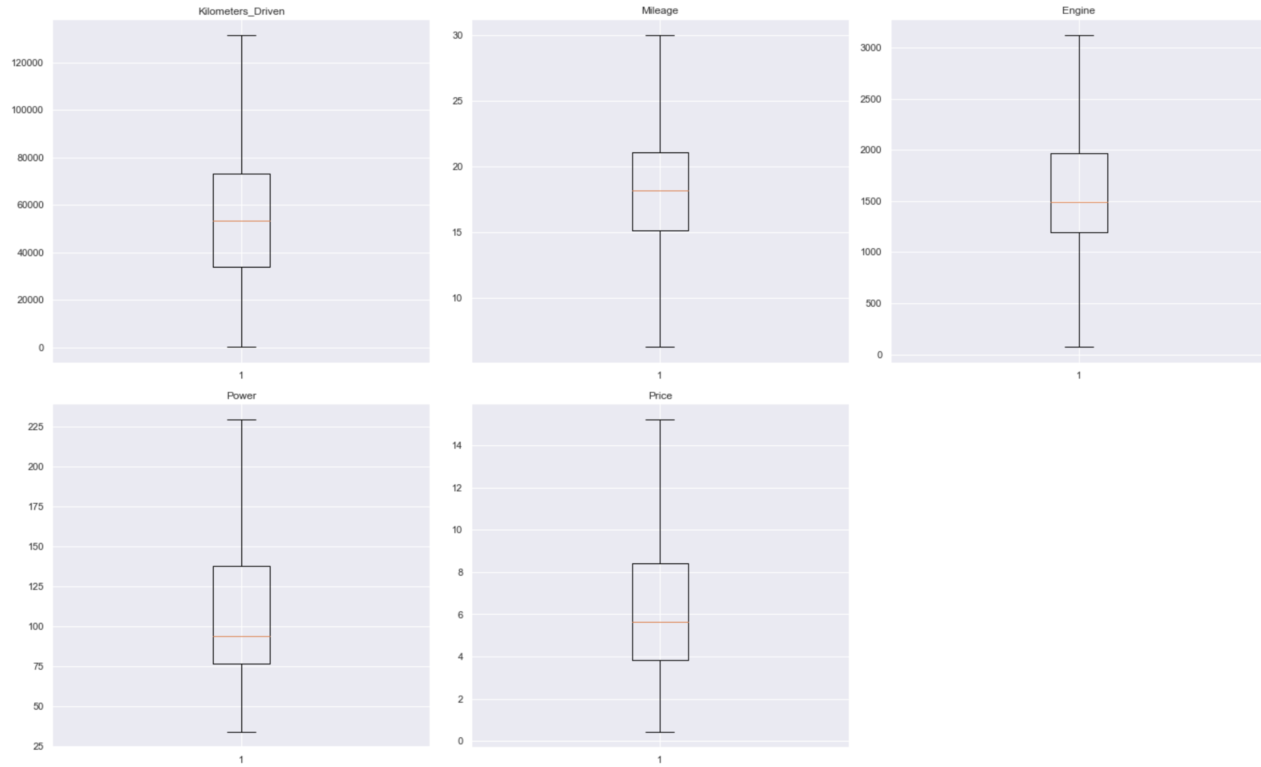
```
# Lets treat outliers by flooring and capping
def treat_outliers(df,col):
    '''
    treats outliers in a varaible
    col: str, name of the numerical varaible
    df: data frame
    col: name of the column
    '''
    Q1=df[col].quantile(0.25) # 25th quantile
    Q3=df[col].quantile(0.75)  # 75th quantile
    IQR=Q3-Q1
    Lower_Whisker = Q1 - 1.5*IQR
    Upper_Whisker = Q3 + 1.5*IQR
    df[col] = np.clip(df[col], Lower_Whisker, Upper_Whisker) # all the values smaller t
                                                            # and all the values above
    return df

def treat_outliers_all(df, col_list):
    '''
    treat outlier in all numerical varaibles
    col_list: list of numerical varaibles
    df: data frame
    '''
    for c in col_list:
        df = treat_outliers(df,c)

    return df
```

In [245...

```
numerical_col = df.select_dtypes(include=np.number).columns.tolist()
df = treat_outliers_all(df,numerical_col)
```

In [245...

```
# lets look at box plot to see if outliers has been treated or not
plt.figure(figsize=(20,30))

for i, variable in enumerate(numeric_columns):
                     plt.subplot(5,3,i+1)
                     plt.boxplot(df[variable],whis=1.5)
                     plt.tight_layout()
                     plt.title(variable)

plt.show()
```



In [245…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               7253 non-null   category
 1   Location           7253 non-null   category
 2   Year               7253 non-null   category
 3   Kilometers_Driven  7253 non-null   int64
 4   Fuel_Type          7253 non-null   category
 5   Transmission       7253 non-null   category
 6   Owner_Type         7253 non-null   category
 7   Mileage            7253 non-null   float64
 8   Engine             7253 non-null   float64
 9   Power              7253 non-null   float64
 10  Seats              7253 non-null   category
 11  Price              7253 non-null   float64
 12  NameMake           7253 non-null   category
dtypes: category(8), float64(4), int64(1)
memory usage: 446.7 KB
```

In [ ]:

Model Building

In [245…
```
df.head()
```

Out[245…

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Eng
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.60	9

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Eng
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67	15
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.20	11
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77	12
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.20	19

In [246…

```
#Defining X and y variables
#X = df.drop(['Price' , 'Name' , 'Year' ] , axis=1)
X = df.drop(['Price' , 'Name' , 'Year' ] , axis=1)
y = df[['Price']]

print(X.head())
print(y.head())
```

```
     Location  Kilometers_Driven Fuel_Type Transmission Owner_Type  Mileage  \
0      Mumbai              72000       CNG       Manual      First    26.60
1        Pune              41000    Diesel       Manual      First    19.67
2     Chennai              46000    Petrol       Manual      First    18.20
3     Chennai              87000    Diesel       Manual      First    20.77
4  Coimbatore              40670    Diesel    Automatic     Second    15.20

   Engine   Power  Seats NameMake
0   998.0   58.16    5.0   MARUTI
1  1582.0  126.20    5.0  HYUNDAI
2  1199.0   88.70    5.0    HONDA
3  1248.0   88.76    7.0   MARUTI
4  1968.0  140.80    5.0     AUDI
    Price
0   1.750
1  12.500
2   4.500
3   6.000
4  15.225
```

In [246…

```
print(X.shape)
print(y.shape)
```

```
(7253, 10)
(7253, 1)
```

Create Dummy Variables

In [246…

```
#X = pd.get_dummies(X, columns=['Location' , 'Year' , 'Fuel_Type' , 'Transmission' , 'O
#X = pd.get_dummies(X, columns=['Location' , 'YearRange' , 'Fuel_Type' , 'Transmission'
X = pd.get_dummies(X, columns=['Location' , 'Fuel_Type' , 'Transmission' , 'Owner_Type'
X.head()
```

Out[246…

	Kilometers_Driven	Mileage	Engine	Power	Location_Bangalore	Location_Chennai	Location_Coimbat

	Kilometers_Driven	Mileage	Engine	Power	Location_Bangalore	Location_Chennai	Location_Coimbat
0	72000	26.60	998.0	58.16	0	0	
1	41000	19.67	1582.0	126.20	0	0	
2	46000	18.20	1199.0	88.70	0	1	
3	87000	20.77	1248.0	88.76	0	1	
4	40670	15.20	1968.0	140.80	0	0	

5 rows × 61 columns

Split the data into train and test

In [246...

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4
```

In [246...

```
X_train.head()
```

Out[246...

	Kilometers_Driven	Mileage	Engine	Power	Location_Bangalore	Location_Chennai	Location_Coim
3181	52000	22.50	998.0	67.04	0	1	
952	92056	13.68	2393.0	147.80	0	0	
2266	131500	11.50	2982.0	171.00	0	0	
1504	43775	20.77	1248.0	88.76	0	0	
7213	131500	22.30	1248.0	74.00	0	0	

5 rows × 61 columns

Choose Model, Train and Evaluate in order to create the Model

- Fitting linear model

In [246...

```
from sklearn.linear_model import LinearRegression
linearregression = LinearRegression()
linearregression.fit(X_train, y_train)

print("Intercept of the linear equation:", linearregression.intercept_)
print("\nCoefficients of the equation are:", linearregression.coef_)

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

predictor = linearregression.predict(X_test)
```

```
Intercept of the linear equation: [12.9000378]

Coefficients of the equation are: [[-2.60399340e-05  2.63600808e-02  2.00161382e-04  3.4
3678929e-02
   3.00980403e-01  2.56202220e-02  1.02924132e+00 -4.79912767e-01
   2.98460169e-01  5.16376899e-02  5.63689716e-01 -1.20194810e+00
  -1.01306754e-01  8.90220740e-02  1.21557967e+00  7.44783246e+00
```

```
 -5.68048698e-01 -1.36202852e-01 -1.06403727e+00 -6.08955182e-01
 -7.22446023e-01 -1.80959661e+00 -7.32912550e+00 -8.49796749e+00
 -8.31143409e+00 -7.06660129e+00 -6.74903554e+00 -7.77056972e+00
 -8.99443141e+00 -8.58891833e+00  1.28522985e+00  3.16877950e+00
  6.74399648e-01 -2.47894522e+00 -2.14213332e+00 -1.80680162e+00
 -1.91447855e-01 -1.36160665e+00 -2.48867593e-12 -1.51481418e+00
 -1.17070809e+00 -1.43353519e+00  7.98317105e-01  2.50043128e+00
  3.56524197e+00  3.05142262e+00 -1.73863476e+00 -1.25010502e+00
  9.66162685e-01  3.90397919e+00 -9.06933092e-01 -1.46161522e+00
  3.00907082e-01  1.91100260e+00 -1.27991693e+00 -1.16812464e+00
 -1.77635684e-15 -2.39614142e+00  6.89563031e-01 -1.44085752e+00
  9.26884181e-01]]
```

Evaluate Model Performances

Sum of Squares Regression

The sum of the differences between the Predicted value and the Mean of the Dependant variable (Price). this defines how well the prdicted line fit our data. If SSR (Sum of Squares) is equal to the SST (Sum of Squares Total), It means that the regression model captures all the observed variablility.

In [246…

```
# Mean Absolute Error on test
mean_absolute_error(y_test, predictor)
```

Out[246…

```
1.7407891619761284
```

The mean absolute error (MAE) calculates the residual for every data point, taking only the absolute value of each so that negative and positive residuals do not cancel out. We then take the average of all these residuals. Effectively, MAE describes the typical magnitude of the residuals.

In [246…

```
# RMSE on test data
mean_squared_error(y_test, predictor)**0.5
```

Out[246…

```
2.3635282955244707
```

The root mean square error (RMSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value. And then takes the square root of the value.

Sum of Squares Error (SSE) Onservation

we can choose to use the Mean Absolute Error, since it gives us the lower error margin

R2 (coefficient of determination) regression score function.

In [246…

```
# R2 Squared: on test
r2_score(y_test, predictor)
```

Out[246…

```
0.694998957328048
```

R2 Observation

- Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

- R2 value is 0.69 which means that in this model independent variables are able to explain 69% of variances in dependent variable

Conclusion

- The Training and testing scores are around 74% and both scores are comparable, hence the model is a good fit.
- R2 Score is 0.69, that explains 69% of total variation in the dataset. So, overall the model is very satisfactory.

Model Statistics

Ordinarey Least Squares (OLS)

In [246…

```
# Lets us build linear regression model using statsmodel

X = sm.add_constant(X)
X_train1, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=

olsmod0 = sm.OLS(y_train, X_train1)
olsres0 = olsmod0.fit()
print(olsres0.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.682
Model:                            OLS   Adj. R-squared:                  0.678
Method:                 Least Squares   F-statistic:                     185.5
Date:                Sat, 17 Apr 2021   Prob (F-statistic):               0.00
Time:                        02:48:23   Log-Likelihood:                -11565.
No. Observations:                5077   AIC:                         2.325e+04
Df Residuals:                    5018   BIC:                         2.363e+04
Df Model:                          58
Covariance Type:            nonrobust
==========================================================================================
=====
                                coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------------------
-----
const                        12.4700      2.425      5.143      0.000       7.716       1
7.224
Kilometers_Driven         -2.604e-05   1.37e-06    -19.003      0.000   -2.87e-05    -2.3
4e-05
Mileage                       0.0264      0.014      1.866      0.062      -0.001
0.054
Engine                        0.0002      0.000      1.099      0.272      -0.000
0.001
Power                         0.0344      0.002     16.536      0.000       0.030
0.038
Location_Bangalore            0.3010      0.220      1.369      0.171      -0.130
0.732
Location_Chennai              0.0256      0.209      0.123      0.902      -0.384
0.436
Location_Coimbatore           1.0292      0.200      5.153      0.000       0.638
1.421
Location_Delhi               -0.4799      0.204     -2.353      0.019      -0.880       -
0.080
Location_Hyderabad            0.2985      0.198      1.504      0.133      -0.091
```

```
0.688
Location_Jaipur               0.0516      0.215      0.240      0.810      -0.370
0.473
Location_Kochi                0.5637      0.200      2.818      0.005       0.172
0.956
Location_Kolkata             -1.2019      0.206     -5.828      0.000      -1.606      -
0.798
Location_Mumbai              -0.1013      0.197     -0.515      0.607      -0.487
0.284
Location_Pune                 0.0890      0.202      0.440      0.660      -0.308
0.486
Fuel_Type_Diesel              1.2156      0.370      3.284      0.001       0.490
1.941
Fuel_Type_Electric            7.4478      1.734      4.296      0.000       4.049      1
0.847
Fuel_Type_LPG                -0.5680      0.808     -0.703      0.482      -2.153
1.017
Fuel_Type_Petrol             -0.1362      0.374     -0.364      0.716      -0.869
0.597
Transmission_Manual          -1.0640      0.113     -9.445      0.000      -1.285      -
0.843
Owner_Type_Fourth & Above    -0.6090      0.801     -0.760      0.447      -2.180
0.962
Owner_Type_Second            -0.7224      0.097     -7.482      0.000      -0.912      -
0.533
Owner_Type_Third             -1.8096      0.251     -7.199      0.000      -2.302      -
1.317
Seats_2.0                    -7.3291      2.529     -2.899      0.004     -12.286      -
2.372
Seats_4.0                    -8.4980      2.423     -3.507      0.000     -13.248      -
3.747
Seats_5.0                    -8.3114      2.408     -3.452      0.001     -13.032      -
3.591
Seats_6.0                    -7.0666      2.452     -2.882      0.004     -11.874      -
2.259
Seats_7.0                    -6.7490      2.409     -2.801      0.005     -11.472      -
2.026
Seats_8.0                    -7.7706      2.415     -3.218      0.001     -12.505      -
3.036
Seats_9.0                    -8.9944      2.936     -3.064      0.002     -14.750      -
3.239
Seats_10.0                   -8.5889      2.592     -3.313      0.001     -13.671      -
3.507
NameMake_AUDI                 1.7152      0.257      6.672      0.000       1.211
2.219
NameMake_BENTLEY              3.5988      2.325      1.548      0.122      -0.959
8.156
NameMake_BMW                  1.1044      0.265      4.171      0.000       0.585
1.623
NameMake_CHEVROLET           -2.0489      0.306     -6.689      0.000      -2.649      -
1.448
NameMake_DATSUN              -1.7121      0.702     -2.439      0.015      -3.089      -
0.336
NameMake_FIAT                -1.3768      0.500     -2.752      0.006      -2.357      -
0.396
NameMake_FORCE                0.2386      1.640      0.145      0.884      -2.976
3.454
NameMake_FORD                -0.9316      0.249     -3.745      0.000      -1.419      -
0.444
NameMake_HINDUSTAN        -1.804e-14   4.66e-15     -3.874      0.000   -2.72e-14   -8.9
1e-15
NameMake_HONDA               -1.0848      0.224     -4.854      0.000      -1.523      -
0.647
NameMake_HYUNDAI             -0.7407      0.213     -3.472      0.001      -1.159      -
0.323
```

```
NameMake_ISUZU                 -1.0035      1.172     -0.856      0.392      -3.302
1.295
NameMake_JAGUAR                 1.2283      0.461      2.664      0.008       0.324
2.132
NameMake_JEEP                   2.9304      0.630      4.650      0.000       1.695
4.166
NameMake_LAMBORGHINI            3.9952      2.444      1.635      0.102      -0.796
8.786
NameMake_LAND                   3.4814      0.406      8.575      0.000       2.685
4.277
NameMake_MAHINDRA              -1.3086      0.269     -4.865      0.000      -1.836      -
0.781
NameMake_MARUTI                -0.8201      0.222     -3.693      0.000      -1.255      -
0.385
NameMake_MERCEDES-BENZ          1.3962      0.245      5.707      0.000       0.917
1.876
NameMake_MINI                   4.3340      0.577      7.512      0.000       3.203
5.465
NameMake_MITSUBISHI            -0.4769      0.545     -0.875      0.381      -1.545
0.591
NameMake_NISSAN                -1.0316      0.325     -3.172      0.002      -1.669      -
0.394
NameMake_OPELCORSA              0.7309      2.306      0.317      0.751      -3.791
5.253
NameMake_PORSCHE                2.3410      0.716      3.271      0.001       0.938
3.744
NameMake_RENAULT               -0.8499      0.294     -2.887      0.004      -1.427      -
0.273
NameMake_SKODA                 -0.7381      0.270     -2.737      0.006      -1.267      -
0.209
NameMake_SMART               1.775e-16   3.21e-16      0.553      0.581   -4.52e-16    8.0
7e-16
NameMake_TATA                  -1.9661      0.277     -7.087      0.000      -2.510      -
1.422
NameMake_TOYOTA                 1.1196      0.246      4.549      0.000       0.637
1.602
NameMake_VOLKSWAGEN            -1.0109      0.241     -4.186      0.000      -1.484      -
0.537
NameMake_VOLVO                  1.3569      0.580      2.339      0.019       0.220
2.494
==============================================================================
Omnibus:                      432.478   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1150.043
Skew:                          -0.483   Prob(JB):                    1.87e-250
Kurtosis:                       5.122   Cond. No.                     1.13e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The smallest eigenvalue is 1.61e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Observation

- P-Value of a variable indicates if the variable is significant or not. If we consider significance level to be 0.05 (5%) than any variable with p-values less than 0.05 would be considered significant.
- In this case Kilometers_Driven , Mileage , Engine , Power , Transmission . Seats are significant
- NameMake has P-Value lower than 0.05 in most categories, but in some categories it is above
- But these variables might contain Multicollinearity which affects the p values, so we first need to deal with multicollinearity and then look for p values

Interpreting the Regression Results:

1. **Adjusted. R-squared**: It reflects the fit of the model.
   - R-squared values range from 0 to 1, where a higher value generally indicates a better fit, assuming certain conditions are met.
   - In our case, the value for Adj. R-squared is **0.69**, which is good.
2. **const coefficient** is the Y-intercept.

   - It means that if all the dependent variables (features: like Country, status, Adult mortality and so on..) coefficients are zero, then the expected output (i.e., the Y) would be equal to the const coefficient.
   - In our case, the value for const coeff is **12.47**
3. **Schooling coeff**: It represents the change in the output Y due to a change of one unit in the Schooling (everything else held constant).

4. **std err**: It reflects the level of accuracy of the coefficients.
   - The lower it is, the higher is the level of accuracy.
5. **P >|t|**: It is p-value.

   - Pr(>|t|) : For each independent feature there is a null hypothesis and alternate hypothesis

     Ho : Independent feature is not significant

     Ha : Independent feature is that it is significant

Pr(>|t|) gives P-value for each independent feature to check that null hypothesis. we are considering 0.05 (5%) as significance level

- A p-value of less than 0.05 is considered to be statistically significant.

1. **Confidence Interval**: It represents the range in which our coefficients are likely to fall (with a likelihood of 95%).

Linear Regression Assumptions

- No Multicollinearity
- Mean of residuals should be 0
- No Heteroscedacity
- Linearity of variables
- Normality of error terms

TEST FOR MULTICOLLINEARITY

- Multicollinearity occurs when predictor variables in a regression model are correlated. This correlation is a problem because predictor variables should be independent. If the correlation between variables is high, it can cause problems when we fit the model and interpret the results. When we have multicollinearity the linear model, The coefficients that the model suggests are unreliable.

- There are different ways of detecting(or testing) multi-collinearity, one such way is Variation Inflation Factor.

- **Variance Inflation factor**: Variance inflation factors measure the inflation in the variances of the regression parameter estimates due to collinearities that exist among the predictors. It is a measure of how much the variance of the estimated regression coefficient βkis “inflated”by the existence of correlation among the predictor variables in the model.

- General Rule of thumb: If VIF is 1 then there is no correlation among the kth predictor and the remaining predictor variables, and hence the variance of βk is not inflated at all. Whereas if VIF exceeds 5 or is close to exceeding 5, we say there is moderate VIF and if it is 10 or exceeding 10, it shows signs of high multi-collinearity.

In [247…

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_series1 = pd.Series([variance_inflation_factor(X.values,i) for i in range(X.shape[1
vif_series1
```

Out[247…

```
const                     15110.878736
Kilometers_Driven             1.520065
Mileage                       3.329591
Engine                        9.784801
Power                         8.434474
                                ...
NameMake_SMART                2.093738
NameMake_TATA               224.418587
NameMake_TOYOTA             478.514742
NameMake_VOLKSWAGEN         359.745538
NameMake_VOLVO               29.352954
Length: 62, dtype: float64
```

- Power and Engine have a VIF score of much greater than 5
- clearly these 2 variables are correlated with each other
- Some of the NameMake values are also highly correlated, we will test be also dropping the NameMake column

Removing Multicollinearity

- To remove multicollinearity
  1. Drop every column one by one, that has VIF score greater than 5.
  2. Look at the adjusted R square of all these models
  3. Drop the Variable that makes least change in Adjusted-R square
  4. Check the VIF Scores again
  5. Continue till you get all VIF scores under 5

In [247…

```
# we drop the one with the highest vif values and check the Adjusted-R Squared
X_train2 = X_train1.drop( 'Engine' , axis=1)
vif_series2 = pd.Series([variance_inflation_factor(X_train2.values,i) for i in range(X_
print('Series before feature selection: \n\n{}\n'.format(vif_series2))
```

```
Series before feature selection:
```

```
const                   0.000000
Kilometers_Driven       1.506915
Mileage                 2.912170
Power                   4.801027
Location_Bangalore      2.486856
                          ...
NameMake_SMART               NaN
NameMake_TATA                inf
NameMake_TOYOTA              inf
NameMake_VOLKSWAGEN          inf
NameMake_VOLVO               inf
Length: 61, dtype: float64
```

In [247…

```
olsmod1 = sm.OLS(y_train, X_train2)
olsres1 = olsmod1.fit()
print(olsres1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.682
Model:                            OLS   Adj. R-squared:                  0.678
Method:                 Least Squares   F-statistic:                     188.8
Date:                Sat, 17 Apr 2021   Prob (F-statistic):               0.00
Time:                        02:48:27   Log-Likelihood:                -11565.
No. Observations:                5077   AIC:                         2.325e+04
Df Residuals:                    5019   BIC:                         2.363e+04
Df Model:                          57
Covariance Type:            nonrobust
==========================================================================================
=====
                                coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------------------
-----
const                        13.0487      2.367      5.513      0.000       8.408       1
7.689
Kilometers_Driven         -2.595e-05   1.37e-06    -18.971      0.000   -2.86e-05    -2.3
3e-05
Mileage                       0.0206      0.013      1.572      0.116      -0.005
0.046
Power                         0.0358      0.002     22.650      0.000       0.033
0.039
Location_Bangalore            0.2984      0.220      1.357      0.175      -0.133
0.729
Location_Chennai              0.0271      0.209      0.129      0.897      -0.383
0.437
Location_Coimbatore           1.0301      0.200      5.157      0.000       0.639
1.422
Location_Delhi               -0.4813      0.204     -2.360      0.018      -0.881       -
0.082
Location_Hyderabad            0.3008      0.198      1.515      0.130      -0.088
0.690
Location_Jaipur               0.0507      0.215      0.235      0.814      -0.371
0.473
Location_Kochi                0.5644      0.200      2.821      0.005       0.172
0.957
Location_Kolkata             -1.2014      0.206     -5.826      0.000      -1.606       -
0.797
Location_Mumbai              -0.0999      0.197     -0.508      0.612      -0.485
0.286
Location_Pune                 0.0903      0.202      0.446      0.656      -0.307
0.487
Fuel_Type_Diesel              1.2255      0.370      3.311      0.001       0.500
```

```
1.951
Fuel_Type_Electric             7.3521      1.732      4.246      0.000       3.957      1
0.747
Fuel_Type_LPG                 -0.5970      0.808     -0.739      0.460      -2.181
0.987
Fuel_Type_Petrol              -0.1743      0.372     -0.468      0.640      -0.904
0.555
Transmission_Manual           -1.0638      0.113     -9.443      0.000      -1.285      -
0.843
Owner_Type_Fourth & Above     -0.6361      0.801     -0.794      0.427      -2.206
0.934
Owner_Type_Second             -0.7219      0.097     -7.476      0.000      -0.911      -
0.533
Owner_Type_Third              -1.8088      0.251     -7.196      0.000      -2.302      -
1.316
Seats_2.0                     -7.5653      2.519     -3.003      0.003     -12.504      -
2.626
Seats_4.0                     -8.8193      2.406     -3.666      0.000     -13.535      -
4.103
Seats_5.0                     -8.6405      2.389     -3.616      0.000     -13.325      -
3.956
Seats_6.0                     -7.3263      2.441     -3.002      0.003     -12.111      -
2.541
Seats_7.0                     -7.0325      2.395     -2.936      0.003     -11.728      -
2.337
Seats_8.0                     -8.0436      2.402     -3.348      0.001     -12.753      -
3.334
Seats_9.0                     -9.2318      2.928     -3.153      0.002     -14.972      -
3.492
Seats_10.0                    -8.8256      2.583     -3.416      0.001     -13.890      -
3.761
NameMake_AUDI                  1.7065      0.257      6.641      0.000       1.203
2.210
NameMake_BENTLEY               3.7072      2.323      1.596      0.111      -0.846
8.261
NameMake_BMW                   1.1101      0.265      4.194      0.000       0.591
1.629
NameMake_CHEVROLET            -2.0542      0.306     -6.707      0.000      -2.655      -
1.454
NameMake_DATSUN               -1.7051      0.702     -2.429      0.015      -3.081      -
0.329
NameMake_FIAT                 -1.3949      0.500     -2.790      0.005      -2.375      -
0.415
NameMake_FORCE                 0.2547      1.640      0.155      0.877      -2.960
3.470
NameMake_FORD                 -0.9091      0.248     -3.667      0.000      -1.395      -
0.423
NameMake_HINDUSTAN         -3.108e-14   9.69e-15     -3.207      0.001   -5.01e-14   -1.2
1e-14
NameMake_HONDA                -1.0686      0.223     -4.791      0.000      -1.506      -
0.631
NameMake_HYUNDAI              -0.7415      0.213     -3.476      0.001      -1.160      -
0.323
NameMake_ISUZU                -0.8915      1.168     -0.763      0.445      -3.181
1.398
NameMake_JAGUAR                1.2526      0.461      2.719      0.007       0.350
2.156
NameMake_JEEP                  2.9005      0.630      4.607      0.000       1.666
4.135
NameMake_LAMBORGHINI           4.0129      2.444      1.642      0.101      -0.778
8.804
NameMake_LAND                  3.4758      0.406      8.561      0.000       2.680
4.272
NameMake_MAHINDRA             -1.2829      0.268     -4.787      0.000      -1.808      -
0.758
```

```
NameMake_MARUTI              -0.8185      0.222     -3.685      0.000      -1.254       -
0.383
NameMake_MERCEDES-BENZ        1.4101      0.244      5.772      0.000       0.931
1.889
NameMake_MINI                 4.3365      0.577      7.516      0.000       3.205
5.468
NameMake_MITSUBISHI          -0.4165      0.542     -0.768      0.442      -1.479
0.646
NameMake_NISSAN              -1.0054      0.324     -3.100      0.002      -1.641       -
0.370
NameMake_OPELCORSA            0.7405      2.306      0.321      0.748      -3.781
5.262
NameMake_PORSCHE              2.4431      0.710      3.443      0.001       1.052
3.834
NameMake_RENAULT             -0.8460      0.294     -2.874      0.004      -1.423       -
0.269
NameMake_SKODA               -0.7147      0.269     -2.658      0.008      -1.242       -
0.188
NameMake_SMART             -9.73e-17   2.42e-17     -4.025      0.000   -1.45e-16    -4.9
9e-17
NameMake_TATA                -1.9577      0.277     -7.059      0.000      -2.501       -
1.414
NameMake_TOYOTA               1.1876      0.238      4.986      0.000       0.721
1.655
NameMake_VOLKSWAGEN          -1.0000      0.241     -4.145      0.000      -1.473       -
0.527
NameMake_VOLVO                1.3173      0.579      2.275      0.023       0.182
2.452
==============================================================================
Omnibus:                      433.940   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1153.489
Skew:                          -0.484   Prob(JB):                    3.33e-251
Kurtosis:                       5.125   Cond. No.                     1.13e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

* Earlier R-squared was 0.69, now it is reduced to 0.68

- We will try again droping Locations and Location

In [247…

```
# we drop the one with the highest vif values

#Location_Bangalore            0.2984      0.220      1.357      0.175      -0.133
#Location_Chennai              0.0271      0.209      0.129      0.897      -0.383
##Location_Coimbatore           1.0301      0.200      5.157      0.000       0.639
#Location_Delhi               -0.4813      0.204     -2.360      0.018      -0.881
#Location_Hyderabad            0.3008      0.198      1.515      0.130      -0.088
#Location_Jaipur               0.0507      0.215      0.235      0.814      -0.371
#Location_Kochi                0.5644      0.200      2.821      0.005       0.172
#Location_Kolkata             -1.2014      0.206     -5.826      0.000      -1.606
#Location_Mumbai              -0.0999      0.197     -0.508      0.612      -0.485

X_train3_1 = X_train2.drop('Location_Bangalore', axis=1)
X_train3_2 = X_train3_1.drop('Location_Chennai', axis=1)
X_train3_3 = X_train3_2.drop('Location_Hyderabad', axis=1)
X_train3_4 = X_train3_3.drop('Location_Jaipur', axis=1)
X_train3_5 = X_train3_4.drop('Location_Mumbai', axis=1)
X_train3_6 = X_train3_5.drop('Location_Pune', axis=1)
```

```python
vif_series3 = pd.Series([variance_inflation_factor(X_train3_6.values,i) for i in range(
print('Series before feature selection: \n\n{}\n'.format(vif_series3))
```

```
Series before feature selection:

const                          0.000000
Kilometers_Driven              1.418936
Mileage                        2.907885
Power                          4.776570
Location_Coimbatore            1.106144
Location_Delhi                 1.073079
Location_Kochi                 1.109358
Location_Kolkata               1.125215
Fuel_Type_Diesel              30.550157
Fuel_Type_Electric             1.061927
Fuel_Type_LPG                  1.262953
Fuel_Type_Petrol              30.883566
Transmission_Manual            2.318495
Owner_Type_Fourth & Above      1.017130
Owner_Type_Second              1.108901
Owner_Type_Third               1.069423
Seats_2.0                     11.227331
Seats_4.0                     88.646876
Seats_5.0                    700.226622
Seats_6.0                     32.523227
Seats_7.0                    511.250314
Seats_8.0                    122.727303
Seats_9.0                      3.033595
Seats_10.0                     7.081827
NameMake_AUDI                       inf
NameMake_BENTLEY                    inf
NameMake_BMW                        inf
NameMake_CHEVROLET                  inf
NameMake_DATSUN                     inf
NameMake_FIAT                       inf
NameMake_FORCE                      inf
NameMake_FORD                       inf
NameMake_HINDUSTAN                  NaN
NameMake_HONDA                      inf
NameMake_HYUNDAI                    inf
NameMake_ISUZU                      inf
NameMake_JAGUAR                     inf
NameMake_JEEP                       inf
NameMake_LAMBORGHINI                inf
NameMake_LAND                       inf
NameMake_MAHINDRA                   inf
NameMake_MARUTI                     inf
NameMake_MERCEDES-BENZ              inf
NameMake_MINI                       inf
NameMake_MITSUBISHI                 inf
NameMake_NISSAN                     inf
NameMake_OPELCORSA                  inf
NameMake_PORSCHE                    inf
NameMake_RENAULT                    inf
NameMake_SKODA                      inf
NameMake_SMART                      NaN
NameMake_TATA                       inf
NameMake_TOYOTA                     inf
NameMake_VOLKSWAGEN                 inf
NameMake_VOLVO                      inf
dtype: float64
```

In [247…

```python
olsmod1 = sm.OLS(y_train, X_train3_6)
```

```
olsres1 = olsmod1.fit()
print(olsres1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.681
Model:                            OLS   Adj. R-squared:                  0.678
Method:                 Least Squares   F-statistic:                     210.5
Date:                Sat, 17 Apr 2021   Prob (F-statistic):               0.00
Time:                        02:48:29   Log-Likelihood:                -11571.
No. Observations:                5077   AIC:                         2.325e+04
Df Residuals:                    5025   BIC:                         2.359e+04
Df Model:                          51
Covariance Type:            nonrobust
=========================================================================================
=====
                             coef    std err          t      P>|t|      [0.025      
0.975]
-----------------------------------------------------------------------------------------
-----
const                     13.2683      2.361      5.619      0.000       8.639      1
7.897
Kilometers_Driven       -2.54e-05   1.33e-06    -19.127      0.000    -2.8e-05    -2.2
8e-05
Mileage                    0.0199      0.013      1.518      0.129      -0.006      
0.046
Power                      0.0356      0.002     22.564      0.000       0.033      
0.039
Location_Coimbatore        0.9437      0.112      8.411      0.000       0.724      
1.164
Location_Delhi            -0.5702      0.119     -4.785      0.000      -0.804      -
0.337
Location_Kochi             0.4813      0.113      4.273      0.000       0.260      
0.702
Location_Kolkata          -1.2826      0.125    -10.296      0.000      -1.527      -
1.038
Fuel_Type_Diesel           1.2755      0.369      3.456      0.001       0.552      
1.999
Fuel_Type_Electric         7.2693      1.731      4.198      0.000       3.875      1
0.664
Fuel_Type_LPG             -0.4881      0.806     -0.606      0.545      -2.068      
1.092
Fuel_Type_Petrol          -0.1526      0.372     -0.411      0.681      -0.881      
0.576
Transmission_Manual       -1.0617      0.113     -9.430      0.000      -1.282      -
0.841
Owner_Type_Fourth & Above -0.6256      0.799     -0.783      0.434      -2.193      
0.942
Owner_Type_Second         -0.7289      0.096     -7.613      0.000      -0.917      -
0.541
Owner_Type_Third          -1.8461      0.249     -7.403      0.000      -2.335      -
1.357
Seats_2.0                 -7.6586      2.520     -3.039      0.002     -12.599      -
2.719
Seats_4.0                 -8.9726      2.405     -3.730      0.000     -13.688      -
4.257
Seats_5.0                 -8.7911      2.389     -3.679      0.000     -13.475      -
4.107
Seats_6.0                 -7.4695      2.441     -3.060      0.002     -12.255      -
2.684
Seats_7.0                 -7.1843      2.395     -2.999      0.003     -11.880      -
2.488
Seats_8.0                 -8.2027      2.402     -3.414      0.001     -12.912      -
3.493
Seats_9.0                 -9.3373      2.927     -3.191      0.001     -15.075      -
```

```
3.600
Seats_10.0                    -9.0138      2.583     -3.490      0.000     -14.077      -
3.951
NameMake_AUDI                  1.7245      0.257      6.712      0.000       1.221
2.228
NameMake_BENTLEY               3.9473      2.322      1.700      0.089      -0.605
8.499
NameMake_BMW                   1.1076      0.265      4.184      0.000       0.589
1.627
NameMake_CHEVROLET            -2.0893      0.306     -6.830      0.000      -2.689      -
1.490
NameMake_DATSUN               -1.6830      0.702     -2.397      0.017      -3.059      -
0.307
NameMake_FIAT                 -1.4332      0.500     -2.868      0.004      -2.413      -
0.454
NameMake_FORCE                 0.2047      1.640      0.125      0.901      -3.010
3.419
NameMake_FORD                 -0.9186      0.248     -3.711      0.000      -1.404      -
0.433
NameMake_HINDUSTAN          9.661e-14   2.41e-14      4.008      0.000    4.93e-14    1.4
4e-13
NameMake_HONDA                -1.0867      0.223     -4.874      0.000      -1.524      -
0.650
NameMake_HYUNDAI              -0.7472      0.213     -3.502      0.000      -1.165      -
0.329
NameMake_ISUZU                -0.9454      1.165     -0.811      0.417      -3.229
1.339
NameMake_JAGUAR                1.2777      0.461      2.774      0.006       0.375
2.181
NameMake_JEEP                  2.8847      0.630      4.582      0.000       1.651
4.119
NameMake_LAMBORGHINI           4.0395      2.445      1.652      0.099      -0.754
8.833
NameMake_LAND                  3.4907      0.406      8.597      0.000       2.695
4.287
NameMake_MAHINDRA             -1.3110      0.268     -4.896      0.000      -1.836      -
0.786
NameMake_MARUTI               -0.8306      0.222     -3.740      0.000      -1.266      -
0.395
NameMake_MERCEDES-BENZ         1.4156      0.244      5.796      0.000       0.937
1.894
NameMake_MINI                  4.3295      0.577      7.503      0.000       3.198
5.461
NameMake_MITSUBISHI           -0.4303      0.541     -0.795      0.427      -1.492
0.631
NameMake_NISSAN               -1.0305      0.324     -3.179      0.001      -1.666      -
0.395
NameMake_OPELCORSA             0.9564      2.306      0.415      0.678      -3.564
5.477
NameMake_PORSCHE               2.4425      0.710      3.442      0.001       1.051
3.834
NameMake_RENAULT              -0.8497      0.294     -2.889      0.004      -1.426      -
0.273
NameMake_SKODA                -0.7410      0.268     -2.761      0.006      -1.267      -
0.215
NameMake_SMART             -4.108e-16   4.05e-16     -1.016      0.310    -1.2e-15    3.8
2e-16
NameMake_TATA                 -1.9642      0.277     -7.083      0.000      -2.508      -
1.421
NameMake_TOYOTA                1.1684      0.238      4.910      0.000       0.702
1.635
NameMake_VOLKSWAGEN           -1.0247      0.241     -4.251      0.000      -1.497      -
0.552
NameMake_VOLVO                 1.3647      0.579      2.357      0.018       0.230
2.500
```

```
==============================================================================
Omnibus:                      418.617   Durbin-Watson:                   1.950
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1086.392
Skew:                          -0.475   Prob(JB):                    1.24e-236
Kurtosis:                       5.058   Cond. No.                     1.13e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

- Drop variables that have P-Value higher than 0.05

Mileage 0.0199 0.013 1.518 0.129 -0.006 0.046 Fuel_Type_LPG -0.4881 0.806 -0.606 0.545 -2.068 1.092 Fuel_Type_Petrol -0.1526 0.372 -0.411 0.681 -0.881 0.576 Owner_Type_Fourth & Above -0.6256 0.799 -0.783 0.434 -2.193 0.942 NameMake_FORCE 0.2047 1.640 0.125 0.901 -3.010 3.419 NameMake_ISUZU -0.9454 1.165 -0.811 0.417 -3.229 1.339 NameMake_LAMBORGHINI 4.0395 2.445 1.652 0.099 -0.754 8.833 NameMake_MITSUBISHI -0.4303 0.541 -0.795 0.427 -1.492 0.631 NameMake_OPELCORSA 0.9564 2.306 0.415 0.678 -3.564 5.477 NameMake_SMART -4.108e-16 4.05e-16 -1.016 0.310 -1.2e-15 3.82e-16

NameMake_DATSUN -1.2946 0.829 -1.561 0.119 -2.920 0.331 NameMake_FORD -0.6055 0.479 -1.265 0.206 -1.544 0.333 NameMake_HONDA -0.7279 0.465 -1.564 0.118 -1.640 0.185 NameMake_HYUNDAI -0.4112 0.463 -0.888 0.375 -1.319 0.497 NameMake_MARUTI -0.4544 0.465 -0.978 0.328 -1.365 0.457 NameMake_NISSAN -0.7034 0.527 -1.334 0.182 -1.737 0.331 NameMake_RENAULT -0.5078 0.509 -0.998 0.318 -1.505 0.489 NameMake_SKODA -0.4235 0.494 -0.858 0.391 -1.391 0.544 NameMake_VOLKSWAGEN -0.7128 0.478 -1.490 0.136 -1.651 0.225

In [247…

```python
X_train4_1 = X_train3_6.drop('Mileage', axis=1)
X_train4_2 = X_train4_1.drop('Fuel_Type_LPG', axis=1)
X_train4_3 = X_train4_2.drop('Fuel_Type_Petrol', axis=1)
X_train4_4 = X_train4_3.drop('Owner_Type_Fourth & Above', axis=1)
X_train4_5 = X_train4_4.drop('NameMake_FORCE', axis=1)
X_train4_6 = X_train4_5.drop('NameMake_ISUZU', axis=1)
X_train4_7 = X_train4_6.drop('NameMake_LAMBORGHINI', axis=1)
X_train4_8 = X_train4_7.drop('NameMake_MITSUBISHI', axis=1)
X_train4_9 = X_train4_8.drop('NameMake_OPELCORSA', axis=1)
X_train4_10 = X_train4_9.drop('NameMake_SMART', axis=1)

X_train4_11 = X_train4_10.drop('NameMake_DATSUN', axis=1)
X_train4_12 = X_train4_11.drop('NameMake_FORD', axis=1)
X_train4_13 = X_train4_12.drop('NameMake_HONDA', axis=1)
X_train4_14 = X_train4_13.drop('NameMake_HYUNDAI', axis=1)
X_train4_15 = X_train4_14.drop('NameMake_MARUTI', axis=1)
X_train4_16 = X_train4_15.drop('NameMake_NISSAN', axis=1)
X_train4_17 = X_train4_16.drop('NameMake_RENAULT', axis=1)
X_train4_18 = X_train4_17.drop('NameMake_SKODA', axis=1)
X_train4_19 = X_train4_18.drop('NameMake_VOLKSWAGEN', axis=1)

vif_series3 = pd.Series([variance_inflation_factor(X_train4_19.values,i) for i in range
print('Series before feature selection: \n\n{}\n'.format(vif_series3))
```

```
Series before feature selection:

const                        5137.990434
```

```
Kilometers_Driven          1.317333
Power                      3.362548
Location_Coimbatore        1.101395
Location_Delhi             1.065565
Location_Kochi             1.103077
Location_Kolkata           1.117661
Fuel_Type_Diesel           1.406884
Fuel_Type_Electric         1.013790
Transmission_Manual        2.245612
Owner_Type_Second          1.097231
Owner_Type_Third           1.048708
Seats_2.0                 11.111689
Seats_4.0                 88.510480
Seats_5.0                699.630828
Seats_6.0                 32.250158
Seats_7.0                510.706161
Seats_8.0                122.618286
Seats_9.0                  3.031891
Seats_10.0                 7.075846
NameMake_AUDI              1.547916
NameMake_BENTLEY           1.019424
NameMake_BMW               1.770336
NameMake_CHEVROLET         1.019869
NameMake_FIAT              1.006253
NameMake_HINDUSTAN              NaN
NameMake_JAGUAR            1.147498
NameMake_JEEP              1.045473
NameMake_LAND              1.106138
NameMake_MAHINDRA          1.654177
NameMake_MERCEDES-BENZ     1.756339
NameMake_MINI              1.104890
NameMake_PORSCHE           1.107766
NameMake_TATA              1.085918
NameMake_TOYOTA            1.481706
NameMake_VOLVO             1.054794
dtype: float64
```

In [247…

```
olsmod1 = sm.OLS(y_train, X_train4_19)
olsres1 = olsmod1.fit()
print(olsres1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.680
Model:                            OLS   Adj. R-squared:                  0.678
Method:                 Least Squares   F-statistic:                     315.0
Date:                Sat, 17 Apr 2021   Prob (F-statistic):               0.00
Time:                        02:48:29   Log-Likelihood:                -11581.
No. Observations:                5077   AIC:                         2.323e+04
Df Residuals:                    5042   BIC:                         2.346e+04
Df Model:                          34
Covariance Type:            nonrobust
================================================================================
==
                             coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
const                     12.6709      2.391      5.300      0.000       7.984      17.3
57
Kilometers_Driven      -2.602e-05   1.28e-06    -20.328      0.000   -2.85e-05   -2.35e-
05
Power                      0.0343      0.001     25.864      0.000       0.032       0.0
37
```

```
Location_Coimbatore       0.9517      0.112      8.498      0.000       0.732       1.1
71
Location_Delhi           -0.5457      0.119     -4.594      0.000      -0.779      -0.3
13
Location_Kochi            0.4941      0.112      4.398      0.000       0.274       0.7
14
Location_Kolkata         -1.2858      0.124    -10.354      0.000      -1.529      -1.0
42
Fuel_Type_Diesel          1.5297      0.079     19.307      0.000       1.374       1.6
85
Fuel_Type_Electric        7.4446      1.692      4.399      0.000       4.127      10.7
62
Transmission_Manual      -1.0579      0.111     -9.546      0.000      -1.275      -0.8
41
Owner_Type_Second        -0.7302      0.095     -7.666      0.000      -0.917      -0.5
43
Owner_Type_Third         -1.8210      0.247     -7.372      0.000      -2.305      -1.3
37
Seats_2.0                -7.0603      2.507     -2.816      0.005     -11.976      -2.1
45
Seats_4.0                -8.8555      2.404     -3.683      0.000     -13.569      -4.1
42
Seats_5.0                -8.7158      2.389     -3.648      0.000     -13.399      -4.0
32
Seats_6.0                -7.3008      2.431     -3.003      0.003     -12.067      -2.5
34
Seats_7.0                -7.1482      2.395     -2.985      0.003     -11.843      -2.4
54
Seats_8.0                -8.2316      2.402     -3.427      0.001     -12.940      -3.5
23
Seats_9.0                -9.3264      2.926     -3.187      0.001     -15.064      -3.5
89
Seats_10.0               -9.1075      2.582     -3.527      0.000     -14.170      -4.0
45
NameMake_AUDI             2.5832      0.214     12.052      0.000       2.163       3.0
03
NameMake_BENTLEY          4.7873      2.400      1.995      0.046       0.083       9.4
91
NameMake_BMW              2.0163      0.218      9.231      0.000       1.588       2.4
45
NameMake_CHEVROLET       -1.2355      0.244     -5.072      0.000      -1.713      -0.7
58
NameMake_FIAT            -0.6187      0.478     -1.295      0.196      -1.556       0.3
18
NameMake_HINDUSTAN     -1.157e-10    2.7e-11     -4.289      0.000   -1.69e-10   -6.28e-
11
NameMake_JAGUAR           2.1370      0.451      4.734      0.000       1.252       3.0
22
NameMake_JEEP             3.7314      0.628      5.939      0.000       2.500       4.9
63
NameMake_LAND             4.2605      0.383     11.131      0.000       3.510       5.0
11
NameMake_MAHINDRA        -0.5051      0.200     -2.523      0.012      -0.897      -0.1
13
NameMake_MERCEDES-BENZ    2.2566      0.192     11.752      0.000       1.880       2.6
33
NameMake_MINI             5.2179      0.574      9.088      0.000       4.092       6.3
43
NameMake_PORSCHE          3.2624      0.723      4.513      0.000       1.845       4.6
80
NameMake_TATA            -1.1205      0.203     -5.512      0.000      -1.519      -0.7
22
NameMake_TOYOTA           1.9926      0.159     12.549      0.000       1.681       2.3
04
NameMake_VOLVO            2.2125      0.576      3.839      0.000       1.083       3.3
```

```
42
==============================================================================
Omnibus:                      408.925   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1046.054
Skew:                          -0.468   Prob(JB):                    7.12e-228
Kurtosis:                       5.017   Cond. No.                     1.13e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Now no feature has p value greater than 0.05, so we'll consider features in X_train4_19 as the final ones and olsres7 as final model

Observations Now Adjusted R-squared is 0.680, Our model is able to explain 68% of variance that shows model is good. The Adjusted-R squared in Olsres0 it was 68% (Where we considered all the variables) this shows that the variables we dropped were not affecting the model much.

Mean of residuals should be 0

In [247...

```
residual= olsres1.resid
np.mean(residual)
```

Out[247...

```
2.249461510415297e-12
```

- Mean of redisuals is very close to 0.

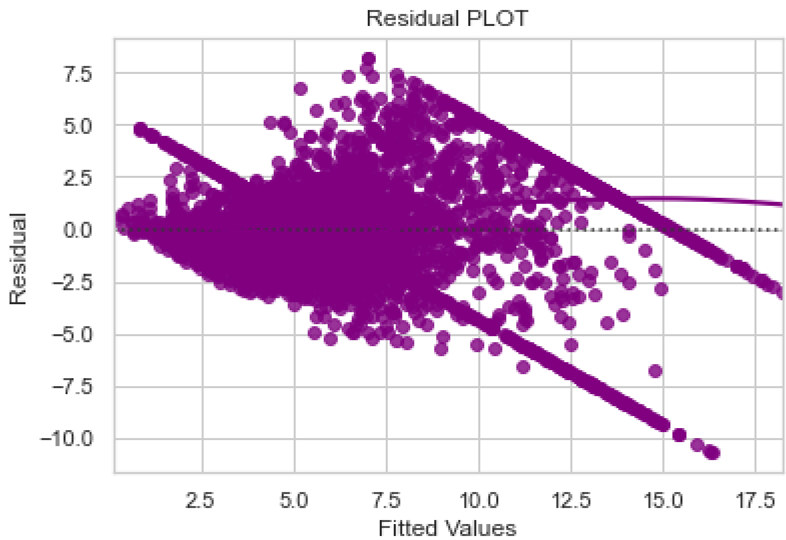
TEST FOR LINEARITY

In [247...

```
residual=olsres1.resid
fitted=olsres1.fittedvalues #predicted values
```

In [247...

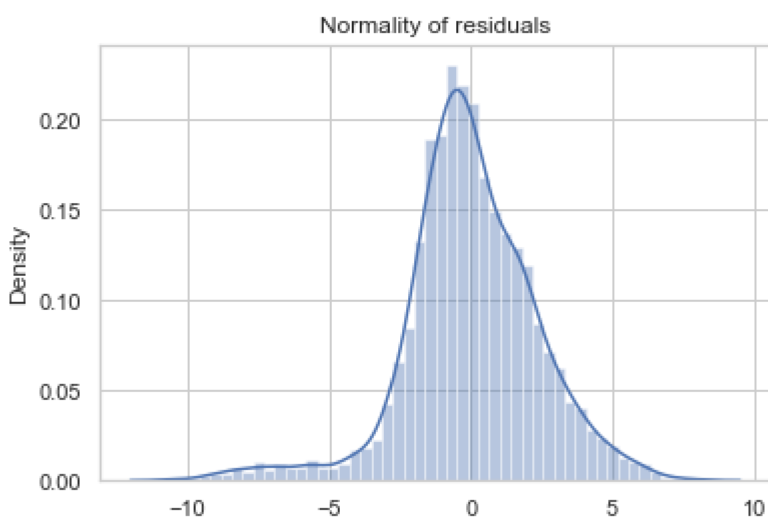
```
sns.set_style("whitegrid")
sns.residplot(fitted,residual,color="purple",lowess=True)
plt.xlabel("Fitted Values")
plt.ylabel("Residual")
plt.title("Residual PLOT")
plt.show()
```



TEST FOR NORMALITY

In [248…

```
sns.distplot(residual)
plt.title('Normality of residuals')
plt.show()
```



In [ ]: